VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Dynamika komplexních sítí
Dynamics of Complex Networks

2015

Lukáš Tomaszek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Tomaszek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Dynamika komplexních sítí
Dynamics of Complex Networks

Zásady pro vypracování:

Cílem této práce je prostudovat oblast dynamiky a evoluce komplexních sítí, vybrat vhodné metody pro popis dynamických vlastností rozsáhlých komplexních struktur a implementovat vybrané metody.

1. Prostudujte teorii v oblasti evoluce dynamických komplexních sítí.
2. Přehled metod týkajících se ohodnocení dynamiky nebo změn v komplexních sítích.
3. Vyberte a popište různě rozsáhlé datové kolekce, nad kterými aplikujete vybrané metody.
4. Implementujte vybrané metody ve formě pluginu do programu Gephi.
5. Porovnejte a analyzujte získané výsledky a vhodně je reprezentujte.

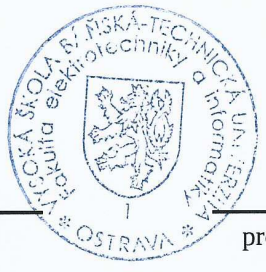
Seznam doporučené odborné literatury:

[1] BELYKH, Igor, et al. Evolving dynamical networks. Physica D: Nonlinear Phenomena, 2014, 267.Complete: 1-6.
[2] HU, Haibo; WANG, Xiaofan. Evolution of a large online social network. Physics Letters A, 2009, 373.12: 1105-1110.
[3] KÖNIG, Michael D.; TESSONE, Claudio J. Network evolution based on centrality. Physical Review E, 2011, 84.5: 056108.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Pavla Dráždilová, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. Hlavně pak vedoucí této práce Mgr. Pavle Dráždilové, PhD. za užitečné poznámky a rady.

Abstrakt

Jak již název napovídá, tato práce se zaměřuje na dynamiku komplexních sítí. Jsou zde popsány základní používané pojmy z teorie grafů, statický pohled na komplexní sítě a hlavně pak dynamický pohled na komplexní sítě pomocí temporálních centralit a change centrality. Tyto centrality jsou pak implementovány jako plugin do nástroje Gephi a jako vlastní nástroj. Vybrané datové kolekce jsou pak analyzovány pomocí implementovaných centralit.

Klíčová slova: komplexní sítě, temporální centrality, change centralita, Gephi

Abstract

As you can see in the title, this thesis is focused on the dynamics of complex networks. There are basic used terms from graph theory, a static view on the complex networks and especially dynamic properties of complex networks are described by temporal centralities and change centrality. These centralities are implemented as Gephi plugin and an own tool. At the end there are analysed some networks using implemented algorithms.

Keywords: complex networks, temporal centralities, change centrality, Gephi

Seznam použitých zkratek a symbolů

DC	–	Degree cetralita
CC	–	Closeness centralita
BC	–	Betweenes centralita
ChC	–	Change centralita
TNC	–	Temporální nejkratší cesta
TCC	–	Temporální closeness centralita
TBC	–	Temporální betweenness centralita

Obsah

1 Úvod

Sítě jsou všude okolo nás. Mnohdy si to ani neuvědomujeme, ale potkáváme se s nimi na každém rohu. Například při cestě do školy či zaměstnání (silniční sítě, železniční sítě), při styku s ostatními lidmi (sítě přátel, sítě lidských kontaktů), při komunikaci (komunikační sítě, emailové sítě) nebo třeba ve volném čase doma brouzdáním na internetu(webové sítě, telekomunikační sítě) a komunikaci na facebooku (sociální sítě). Takovýchto sítí existuje spousta. Sami je vytváříme a měníme. A právě na sítě měnící se v čase je zaměřena tato práce.

Jelikož nás tyto sítě obklopují, chtěli bychom je umět analyzovat a predikovat jejich další vývoj. Dokázali bychom tak například určovat, kde je třeba upravit silnice, aby nedocházelo k dopravním nehodám. Jak tyto silnice upravit, aby nedocházelo k dopravním zácpám. Dále bychom dokázali určovat hrozby mezi lidmi. Může nás zajímat analýza teroristických sítí. Pokud bychom znali intenzitu elektronické komunikace mezi členy takového skupiny, mohli bychom tak předcházet útokům.

Tato práce se zaměřuje na různé centrality komplexních sítí, které se mění v čase. Tyto temporální centrality pak popisují vlastnosti jednotlivých prvků komplexní sítě a umožňují nám najít zajímavé charakteristické či významné objekty v takovéto síti.

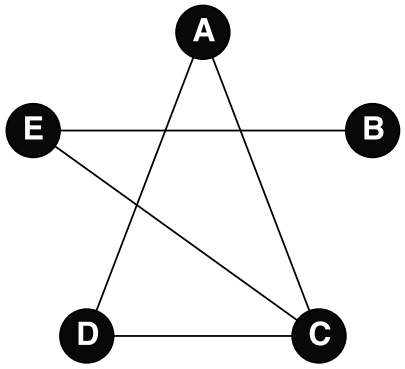
V první kapitole stručně projdeme základní teorii grafů. Jaké typy grafu existují. Jak je můžeme reprezentovat a základní pojmy jako je stupeň, cesta či počet vrcholů a hran. V kapitole 2 se podíváme na statický pohled na komplexní sítě. Tři nejčastější vlastnosti, které se u statických sítí určují a nadefinujeme si tzv. centrality. V další kapitole poté přejdeme od statických sítí k dynamickým. Řekneme si, co je to temporální a časově uspořádaný graf a nadefinujeme si vztahy pro výpočet centralit na dynamických sítích. V páté kapitole poté popíšeme algoritmy, které tyto centrality dokáží spočítat. V následujících dvou kapitolách poté ukážeme základní práci s nástrojem Gephi a popíšeme vstupy, výstupy pro Gephi a vlastní nástroj. V předposlední kapitole ukážeme výsledky získané pomocí naimplementovaných algoritmů. Na konci této práce se podíváme na různé komplikace a problémy, které nastaly.

2 Teorie grafů

Teorie grafů je teoretickým základem komplexních sítí. V této kapitole si nejprve nadefinuje pojem graf a podíváme se na různé typy grafů a jejich odlišnosti. Dále si nadefinujeme základní pojmy z teorie grafů, které se budou dále v této práci nacházet a v neposlední řadě si ukážeme, jak můžeme grafy reprezentovat.

2.1 Graf a typy grafů

Mnozí si pod pojmem graf představí graf ze statistiky, či graf funkce. V diskrétní matematice rozumíme grafem strukturu objektů a vazeb mezi nimi. Ukázku můžeme vidět na následujícím obrázku 1.



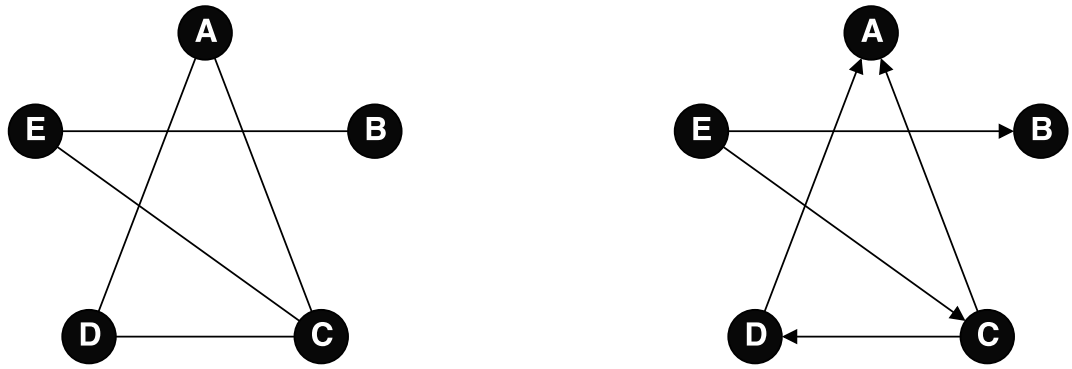
Obrázek 1: Ukázka grafu

Formálně, jak je uvedeno v [14], můžeme říci, že **graf** G (nebo také **jednoduchý graf**) je uspořádaná dvojice $G = (V, E)$, kde V je neprázdná množina vrcholů (někdy také uzlů, bodů) a E je množina hran (někdy také vazeb). E je množina dvouprvkových podmnožin množiny V.

Podle této definice je vazba mezi dvěma vrcholy oboustranná, protože nedokážeme určit ze kterého a do kterého uzlu daná hrana vede. Říkáme, že takový graf je **neorientovaný**. Zavedením směrů pak dokážeme pořadí rozlišit. Tento směr v grafu zaznamenáváme šipkou a mluvíme o tzv. **orientovaném grafu**. Ukázka neorientovaného a orientovaného grafu je na obrázku 2.

Dalším důležitým pojmem, který budeme v této práci potřebovat je **podgraf**. Podgraf vznikne z původního grafu odebráním libovolného počtu hran a vrcholů, avšak nesmí zůstat žádná hrana volná, tedy každá hrana musí spojovat 2 vrcholy. Formálně, podle [14], říkáme, že graf $G_1(V_1, E_1)$ nazýváme podgrafem grafu $G(V, E)$, jestliže $(V_1 \subseteq V) \wedge (E_1 \subseteq E)$.

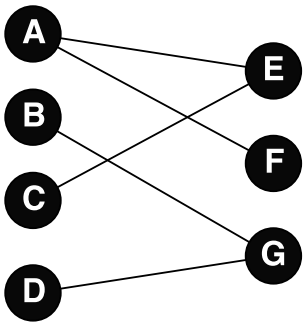
Posledním typem grafu, který si zde ukážeme, je tzv. **bipartitní graf**. Jedná se o typ grafu, kdy vrcholy reprezentují 2 druhy objektů a vazby jsou pouze mezi objekty různého typu. Vrcholy tak tvoří 2 skupiny, kdy hrany jsou pouze mezi skupinami, nikoliv v rámci skupiny. Ukázka takového grafu je na obrázku 3.



(a) Jednoduchý (neorientovaný) graf

(b) Orientovaný graf

Obrázek 2: Ukázky grafů



Obrázek 3: Bipartitní grafu

Skupiny nemusí být pouze 2. Může existovat i několik skupin. Pokud nebudou existovat vazby v rámci těchto skupin, bude se jednat o tzv. **r-partitní graf**, kde r je počet skupin.

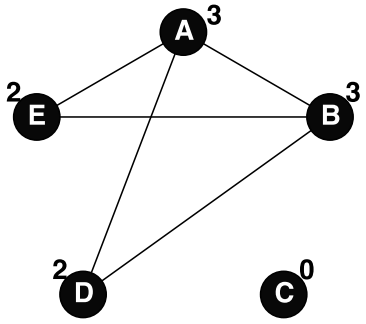
2.2 Základní pojmy z teorie grafů

2.2.1 Stupeň vrcholu (značíme $deg(u)$)

Stupeň vrcholu u označuje, kolik uzlů je s daným vrcholem přímo spojeno hranou. Jedná - li se o orientovaný graf, pak rozlišujeme vstupní a výstupní stupeň vrcholu. Tyto hodnoty značíme $deg_{in}(u)$ a $deg_{out}(u)$ a určují kolik hran vede do (z) daného uzlu u.

Na následujícím obrázku 4 můžeme vidět graf o pěti vrcholech, kdy u každého vrcholu je zobrazen i jeho stupeň. Vrchol A je spojen hranami s vrcholy B, D a E, tudíž je stupeň tohoto vrcholu $deg(A) = 3$. Vrchol B je také spojen s třemi vrcholy a tedy je stejně

jako u vrcholu A stupeň $deg(B) = 3$. Podobně bych mohl postupovat u dalších vrcholů, či jiných grafů.



Obrázek 4: Stupeň vrcholů

2.2.2 Vzdálenost (nejkratší cesta)

Neformálně můžeme říci, že vzdálenost mezi dvěma vrcholy je nejmenší počet hran, které musíme projít, abychom se dostali z jednoho vrcholu do druhého. Formálně, jak je uvedeno v [14], pro definici vzdálenosti musíme nadefinovat nejprve pojem sled.

Sledem v_0v_n v grafu G rozumíme takovou posloupnost vrcholů a hran $(v_0, e_1, v_1, ..., e_n, v_n)$, kde v_i jsou vrcholy grafu G a e_i jsou hrany grafu G, přičemž každá hrana e_i má koncové vrcholy v_{i-1} a v_i. Počet hran nazveme délkou sledu v_0v_n.

Máme-li nadefinován pojem sled, můžeme pak vzdálenost definovat následovně. **Vzdálenost** $d(u, v)$ mezi vrcholy u a v grafu G je dána délkou nejkratšího sledu mezi u a v v G. Pokud sled mezi u, v neexistuje, položíme vzdálenost $d(u, v) = \infty$.

Podíváme-li se zpět na předchozí obrázek 4. Se zavedením této definice vzdálenosti můžeme říci, že vzdálenost mezi vrcholy A a B, A a D, A a E je rovna 1, protože vrchol A je přímo spojen hranou s těmito vrcholy. Vzdálenost mezi vrcholy D a E je rovna 2 a vzdálenost mezi C a libovolných vrcholem je rovna ∞.

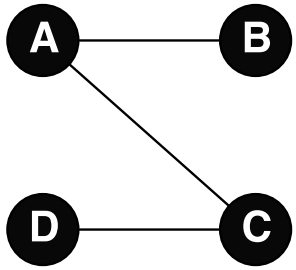
2.2.3 Počet vrcholů, hran

Nyní si ukážeme značení pro počet vrcholů a hran, protože ho budeme v tomto textu dále potřebovat a budeme ho využívat. Počet vrcholů značíme $|V|$ nebo častěji používaným malým písmenem n. Počet hran $|E|$ nebo také m.

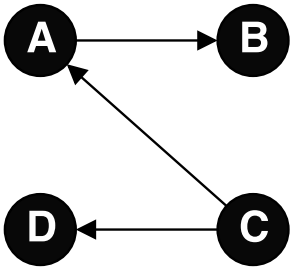
2.3 Reprezentace grafu

Pro zjištění vlastností grafu, a jeho využití ve výpočetní technice, musíme umět graf vhodně reprezentovat pomocí nějaké vhodné struktury. Nyní si ukážeme tři základní možnosti.

Pro jednoduchost si ukážeme všechny možnosti na následujících grafech z obrázku 5 a uvidíme tak rozdíly mezi jednotlivými reprezentacemi.



(a) Jednoduchý (neorientovaný) graf



(b) Orientovaný graf

Obrázek 5: Ukázky grafů

2.3.1 Incidenční matice (značíme I)

První možností reprezentace grafu je pomocí tzv. incidenční matice. V této matici zaznamenáváme, ze kterého do kterého uzlu daná hrana směřuje. Jedná se o matici typu $n \times m$. Je-li daná hrana spojena s daným uzlem, bude se v matici vyskytovat hodnota 1. Není-li tomu tak bude v matici hodnota 0.

Pro orientovaný graf se místo hodnoty 1 využívá hodnot 1 a -1. Máme-li tedy orientovanou hranu z vrcholu u do vrcholu v, bude na pozici u hodnota 1 a na pozici v hodnota -1.

Pro naše příklady budou tedy matice vypadat následovně:

$$I_{neorient} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, I_{orient} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{pmatrix}.$$

2.3.2 Matice sousednosti (značíme A)

Další možností je využití matice sousednosti. Jedná se o matici typu $n \times n$. Jsou-li dva uzly spojeny hranou, je poté hodnota v této matici 1, jinak 0. Tedy pro naše příklady vypadají matice takto:

$$A_{neorient} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, A_{orient} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

2.3.3 Seznam sousedů

Pro malé grafy můžeme použít dva výše zmíněné způsoby reprezentace. Pro větší je však uchování v podobě matice nepoužitelné. Již pro grafy o tisíci uzlech bychom museli uchovávat milión hodnot, což je paměťově náročné. Nemluvě ještě o větších grafech. Jak se však ukázalo, a v dalších kapitolách si to ukážeme, v reálných sítích je často každý uzel spojen pouze s několika málo sousedy. Je tedy výhodnější uchovávat pro každý uzel seznam jeho sousedů. Ukázku našich grafů vidíme v následující tabulce 1.

Vrchol	Sousedé neorientovaný graf	Sousedé orientovaný graf
A	B,C	B
B	A	-
C	A,D	A
D	C	D

Tabulka 1: Seznam sousedů

3 Statický pohled na komplexní sítě

3.1 Vlastnosti

U statických sítí se nejčastěji uvádějí 3 základní vlastnosti [21]. Průměrná délka nejkratší cesty, shlukovací koeficient a stupeň distribuce. Tyto vlastnosti si nyní popíšeme.

3.1.1 Průměrná délka nejkratší cesty

V předchozí kapitole jsme si nadefinovali pojem vzdálenost. Průměrná délka nejkratší cesty je dána průměrem všech vzdáleností mezi všemi vrcholy. Můžeme to zapsat vztahem:

$$d_{AVG} = \frac{\sum_{u}^{n} \sum_{v:v \neq u}^{n} d(u,v)}{n(n-1)}.$$

3.1.2 Shlukovací koeficient

Shlukovací koeficient nám udává, jaká je pravděpodobnost, že dva libovolné uzly, které mají alespoň jednoho společného souseda, jsou také spojeny hranou. Zapsáno vztahem:

$$C_u = \frac{2e_u}{deg(u)(deg(u)-1)},$$

kde e je počet navzájem propojených sousedů uzlu u.

3.1.3 Distribuce stupňů

Distribuce stupňů nám udává, jaká je pravděpodobnost, že daný uzel je právě stupně k a vypočítáme ji tak, že spočítáme počet vrcholů daného stupně a vydělíme počtem všech uzlů.

3.2 Centrality

Další zajímavé vlastnosti komplexních sítí jsou tzv. centrality. Udávají nám, jak moc je který uzel v dané síti důležitý. Popíšeme si zde 3 základní centrality podle [15] a to degree, closeness a betweenness centralitu. Existuje však mnoho dalších centralit, v této práci se však budeme zabývat pouze těmito základními. Další centrality můžeme nalézt např. v článku [7].

Pro lepší porovnávání a práci s výsledky se hodnoty centralit uvádějí normalizované.

Názvy jednotlivých centralit se nebudou překládat. Ani pro statické a následně ani pro dynamické sítě. V české literatuře se většinou tyto názvy nepřekládají a také zde využijeme anglické názvy.

3.2.1 Degree centralita (DC)

Úplně základní centralitou je degree centralita. Ta nám říká, že čím má daný vrchol vyšší stupeň, tím je v síti důležitější. Tedy čím více bude mít daný vrchol sousedů, tím vyšší bude hodnota této centrality. Hodnotu této centrality vypočítáme podle následujícího vztahu:

$$C_u^{deg} = \frac{deg(u)}{n-1},$$

kde $deg(u)$ je stupeň vrcholu, pro který danou centralitu počítáme a n je počet všech vrcholů grafu.

3.2.2 Closeness centralita (CC)

Closeness centralita nám řadí uzly podle toho, jak rychle dokáží komunikovat s ostatními uzly v síti. Tato centralita se vypočítá podle následujícího vztahu:

$$C_u^{clo} = \frac{1}{n-1} \sum_{v \neq u \in V} d_{u,v},$$

kde $d_{u,v}$ je vzdálenost mezi uzly u a v, a je dána jako průměr nejkratších cest z daného uzlu do všech ostatních uzlů.

U sítí, kde se nachází více než jedna komponenta je tento vztah nepraktický. Centralita se počítá pouze v rámci komponenty. Porovnávání hodnot napříč komponentami nemá správnou vypovídající hodnotu. Proto se někdy využívá následující vztah:

$$C_u^{clo} = \frac{1}{n-1} \sum_{v \neq u \in V} \frac{1}{d_{u,v}}.$$

3.2.3 Betweenness centralita (BC)

Tato poslední centralita nám dělí uzly podle toho, v kolika nejkratších cestách se daný uzel nachází. Čím více nejkratších cest daným uzlem prochází, tím je uzel důležitější. Pro výpočet se využívá následujícího vztahu:

$$C_u^{bet} = \sum_{v \neq u, w \neq u \in V} \frac{p_{vw}(u)}{p_{vw}},$$

ve kterém p_{vw} znamená počet cest z uzlu v do uzlu w a $p_{vw}(u)$ je počet cest procházejících uzlem.

4 Dynamický pohled na komplexní sítě

Doposud jsme se dívali na komplexní sítě jako na statické objekty, které mají pevnou strukturu a nemění se. Jak je však uvedeno v článku [5], komplexní sítě jsou sítě s tisíci až milióny uzly, jejichž struktura je nepravidelná, komplexní a dynamicky se rozvíjející v čase. Nyní se podíváme, jak můžeme definovat takovouto síť a ukážeme si některé vlastnosti, které na takovýchto sítích můžeme analyzovat.

Dynamická síť se oproti statické mění. Není pořád stejná. Hrany zde mohou měnit svou váhu, mohou vznikat či zanikat. Nebo se podobně mohou měnit uzly. Podle toho, jak se síť mění, můžeme rozlišovat různé typy změn, jak je uvedeno v článku [11]. Obecně pak můžeme říci, že dynamická síť $G(V, E, f_t, g_t)$ je dána množinou vrcholů V, množinou hran E a dále funkcemi f_t a g_t, kde $f_t : V \longmapsto X$ a $g_t : E \longmapsto Y$. Za X a Y můžeme dosadit libovolnou množinu. Například přirozená čísla. Poté nám u hran bude funkce g_t reprezentovat měnící se váhu dané hrany. Nebo můžeme pomocí těchto funkcí určovat, zda daná hrana (uzel) bude existovat či nikoliv.

Pro analýzu jsou však takovéto sítě celkem problematické. Představme si síť komunikace mezi uživateli pomocí emailů, kde vrcholy reprezentují uživatelé a hrana bude reprezentoval poslaný email mezi dvěma uživateli. Je jasné, že hrana se v grafu zobrazí vždy jen na chvíli a opět zanikne. Zjištění vlastností takovýchto sítí, které se velmi rychle a náhodně mění je pak obtížné. Zavedly se proto tzv. temporální sítě.

4.1 Temporální sítě

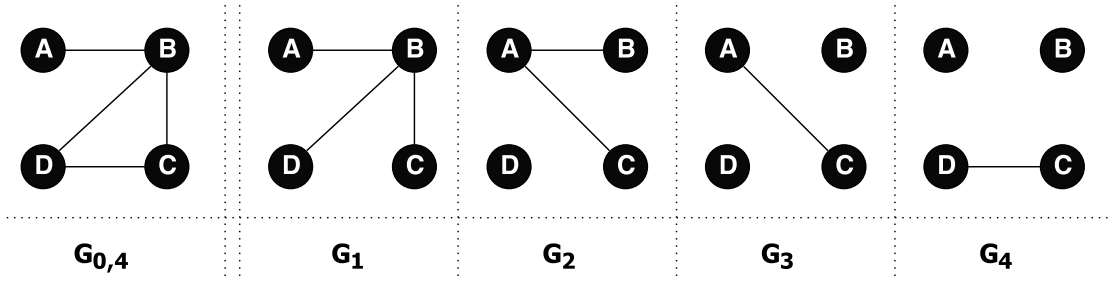
Podobně jako je uvedeno v [18], můžeme temporální síť $G^w_{t_{min},t_{max}}(V, E)$, zkráceně $G_{t_{min},t_{max}}$, definovat jako sekvenci grafů $(G_1(V, E_1), G_2(V, E_2), ..., G_P(V, E_P))$, kde $P = ((t_{max} - t_{min})/w) = |G^w_{t_{min},t_{max}}|$ je počet grafů, w je velikost každého časového okna vyjádřena v nějakých jednotkách (sekundy, hodiny, dny, ...), V je množina uzlů, která je pro všechny grafy shodná a E_i je množina hran grafu G_i. Hrana mezi uzly u a v existuje v grafu G_P právě tehdy, když v intervalu $< t_{min} + w \times (q - 1), t_{min} + w \times q)$, kde $q = 1, ..., P$ existuje alespoň částečně hrana mezi u a v.

Pro ukázku si vytvoříme temporal graf $G_{0,4}$ o čtyřech oknech, který bude mít čtyři vrcholy A, B, C, D a existence hran je dána následující tabulkou 2.

Hrana	Časový interval
(A, B)	(0, 2)
(A, C)	(1, 3)
(B, C)	(0, 1)
(B, D)	(0, 1)
(C, D)	(3, 4)

Tabulka 2: Hrany grafu G

Jak bude výsledný graf vypadat, můžeme vidět na následujícím obrázku 6. Můžeme zde vidět celkový graf $G_{0,4}$, který je statický a mnoho nám o dané síti neřekne. Dále pak můžeme vidět sekvenci grafů G_1, G_2, G_3, G_4 v intervalech $< 0, 1), < 1, 2), < 2, 3), < 3, 4)$.



Obrázek 6: Graf G v jednotlivých časových intervalech

Podobně jako u statických sítí, tak i u dynamických budeme označovat počet vrcholů n a počet hran m. Počet vrcholů se oproti statickým sítím nijak neliší. Počtem hran m budeme myslet počet hran časově orientovaného grafu, který si nyní nadefinuje.

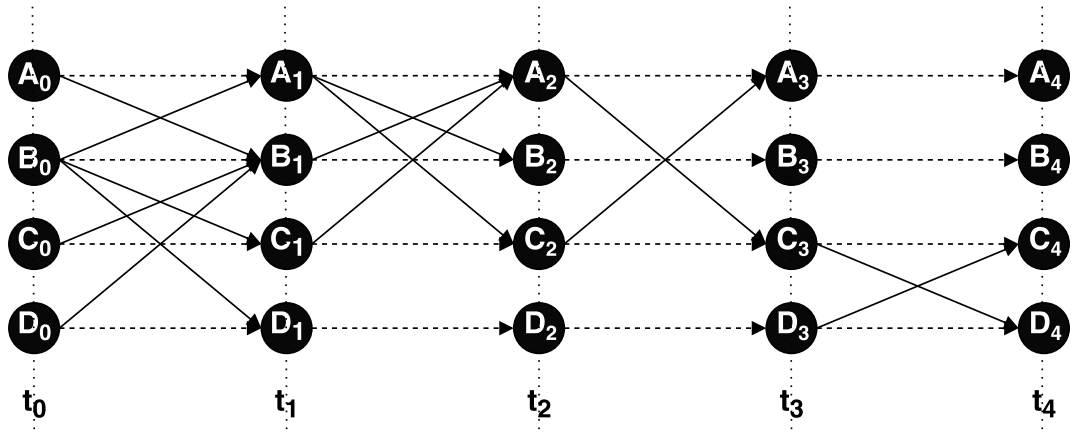
4.2 Časově orientované sítě

Přestože reprezentace temporálních sítí jsou velice intuitivní, není lehké přímo analyzovat temporální vlastnosti dané sítě. Proto si zavedeme poslední typ sítě, tzv. časově orientovanou síť, jak je uvedeno v článku [13]. Jedná se o orientovaný asymetrický graf $G(V, E)$, který obsahuje vrcholy v_t, pro všechny vrcholy v z temporálního grafu a pro všechna $q = 0, 1, ..., P$ a dále obsahuje hrany mezi všemi vrcholy $v_{(q-1)}, v_q$ a hrany mezi vrcholy $u_{(q-1)}, v_q$ pokud existuje hrana v temporálním grafu G_q. Časově orientovaný graf předchozího příkladu můžeme vidět na následujícím obrázku 7.

Jinými slovy můžeme říci, že se jedná o graf, kdy každý vrchol je zobrazen v čase $q = 0, 1, ..., P$ a hrana zde existuje právě tehdy, když existuje alespoň částečně v daném intervalu. Navíc jsou zde spojeny shodné vrcholy, jakoby existovala smyčka v každém vrcholu v každém čase. Můžeme si to představit tak, že procházíme graf, a v každém časovém úseku můžeme projít jednou hranou, ale také můžeme zůstat stát v daném vrcholu. Proto zde existují tyto hrany, které jsou na obrázku 7 vyznačeny čárkovanou čarou.

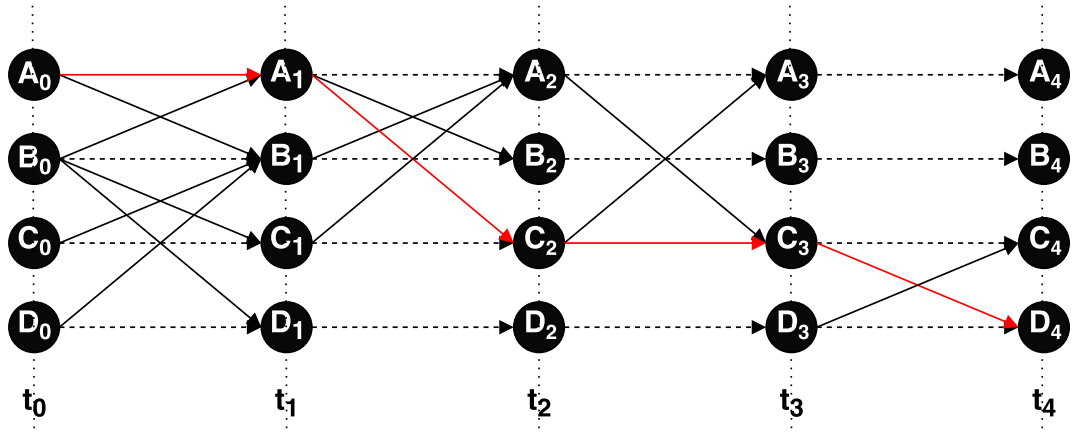
4.3 Temporální nejkratší cesta (TNC)

Podobně jako u statických sítí, tak i u dynamických sítí určujeme nejkratší cestu. Jak je uvedeno v [13], výpočet provádíme na časově uspořádaném grafu a nejkratší cestu d^T z uzlu u do uzlu v v časovém intervalu (r, s) definujeme jako libovolnou cestu $p = (u_r, ..., v_t)$, kde $r < t \leq q$ s délkou $d^T = min_{r<q\leq s} d(u_r, v_q)$, kde $d(u, v)$ je délka nejkratší cesty statického grafu. V případě, že cesta mezi vrcholy u a v neexistuje, položíme vzdálenost $d^T = \infty$.



Obrázek 7: Graf G z obrázku 6 jako časově uspořádaný graf

Na následujícím obrázku 8 můžeme vidět temporální nejkratší cestu, vyznačenu červeně, z vrcholu A do vrcholu D v časovém intervalu (0,4). Tato cesta vede z uzlu A_0 přes uzly A_1, C_2, C_3 do uzlu D_4 a její délka je $d^T = 4$.



Obrázek 8: Temporální nejkratší cesta z uzlu A do uzlu D

4.4 Temporální centrality

Výpočet statických a temporálních centralit se liší pouze v tom, že se počítá s TNC a výpočet se provádí nad všemi časovými okny. Temporální centrality jsou například definované v [13]. Již víme, co daná centralita určuje, proto si nyní ukážeme pouze vzorce, jak dané centrality vypočítat.

4.4.1 Temporální degree centralita (TDC)

Výsledek této centrality je stejný, jako kdybychom se na dynamickou síť podívali v určitých intervalech a považovali ji za statickou a výsledky nakonec zprůměrovali. Tato centralita nám mnoho nového nepřináší a proto je vhodné místo této centrality využít např. change centralitu.

4.4.2 Temporální closeness centralita (TCC)

Vztah pro výpočet TCC vypadá následovně:

$$C_u^{TClo} = \sum_{o \leq p < q} \sum_{v \neq u \in V} \frac{1}{d_{p,q}^T(u,v)}.$$

Sčítáme zde převrácené hodnoty všech TNC z uzlu u do všech ostatních uzlů napříč všemi okny a to tak, že oknu se posouvá pouze počáteční hranice. Pokud bychom chtěli vztah normalizovat je třeba jej ještě vynásobit převrácenou hodnotou počtu cest a oken.

4.4.3 Temporální betweenness centralita (TBC)

Podobně jako u statických sítí je vzorec dán poměrem cest z vrcholu v do vrcholu w procházejících vrcholem u ku všem cestám mezi těmito vrcholy. Pro normalizaci je třeba opět vynásobit převrácenou hodnotou počtu cest a oken. Vztah vypadá následovně:

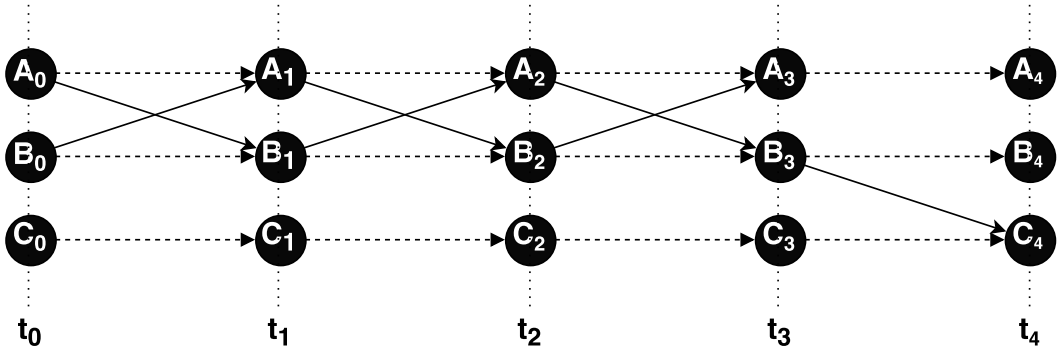
$$C_u^{TBet} = \sum_{o \leq p < q} \sum_{v \neq u \in V} \frac{p_{v,w}^T(u)}{p_{v,w}^T}.$$

Ve většině literatury se normalizuje pomocí počtu cest a okem. V již zmíněné referenci [13] se však normalizuje pomocí počtu existujících nejkratších cest.

Nyní se zkusíme trochu zamyslet nad takto definovanou centralitou. Mějme časově orientovaný graf jako je na následujícím obrázku 9. Vidíme, že vzdálenost mezi vrcholy A a C v časovém intervalu (0, 4) je rovna 4. Počet cest však není tak snadné určit. Nevíme zda cesta A_0, B_1, A_2, B_3, C_4 je shodná s cestou A_0, A_1, B_2, B_3, C_4. Definice neuvádí, zda jsou si tyto cesty rovny, či nikoliv.

Podívejme se tedy na praktické použití. Představme si emailovou síť, kde uzly jsou jednotliví lidé a hrany mezi nimi, budou prezentovat výměnu zprávy. V tomto případě by bylo vhodnější počítat s tím, že dané cesty jsou různé, a tedy je započítat vícekrát, protože v daném čase, měla dotyčná osoba různé údaje, které mohly přijít zprávami. Jinými slovy v čase 0 neměla osoba A žádné údaje z komunikace, kdežto v čase 1 již mohla tato osoba pracovat s údaji, které obdržela od osoby B.

Na druhou stranu si však můžeme představit, že dané uzly představují města a hrana mezi uzly realizuje možná spojení pomocí autobusové dopravy. Lze tedy předpokládat, že pokud budu chtít cestovat mezi uzly A a C, pojedu z A do B a následně z B do C. Určitě se nestane, že pojedu z místa A do místa B, pak se vrátím a opět pojedu tu samou trasu znovu. Tedy počet všech možných cest by se v tomto případě lišil.



Obrázek 9: Ukázka sítě pro TBC

Většina datových kolekcí, které hodlám prozkoumat, je komunikačních, tedy budu pracovat pouze s první variantou a to, že jednotlivé cesty jsou různé, ačkoliv vedou přes stejné uzly, avšak v jiném čase.

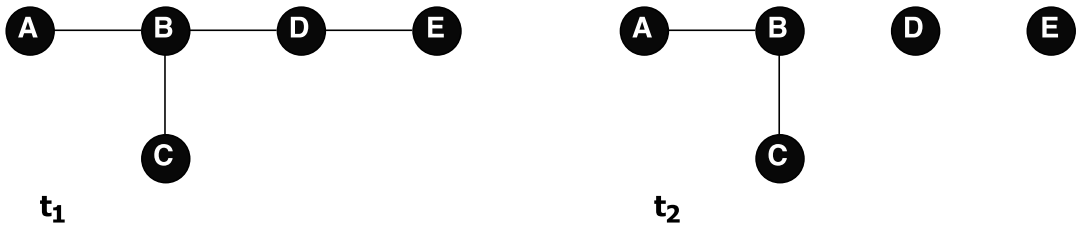
4.5 Change centrality (ChC)

Oproti TCC a TBT není change centralita definování na temporálním grafu, ale na grafu ve dvou časových řezech t_1 a t_2. Potom pro graf G(V, E) v těchto řezech, podle [8], definujeme změnu okolí uzlu u ve vzdálenosti 1:

$$r_{t_1,t_2}(u) = \frac{|N_{t_1}(u)\Delta N_{t_2}(u)|}{|N_{t_1}(u) \cup N_{t_2}(u)|},$$

kde $N_t(u) = \{v \in V : d_t(u, v) = 1\}$, tedy to jsou uzly spojené s uzlem u, v čase t, ve vzdálenosti 1. V čitateli tedy máme změnu v okolí pro daný uzel. Tedy hrany, které existují v okolí uzlu u v čase t_1, ale neexistují v okolí uzlu u v čase t_2 a naopak. Ve jmenovali jsou pak všechny uzly v okolí u existující v čase t_1 nebo t_2.

Nyní se podíváme na příklad. Mějme graf ve dvou časových řezech, jako je na následujícím obrázku 10, a vypočítejme pro něj změnu v okolí velikosti 1.



Obrázek 10: Ukázka sítě pro ChC

Vidíme, že pro uzel A se okolí nezměnilo a uzel byl spojen s uzlem A. Tedy změna ve vzdálenosti 1 je rovna 0 ($\frac{0}{1}$). Uzel B ztratil v čase t_2 vazbu na uzel D a byl spojen s uzly

A, C a D. Tedy jeho změna ve vzdálenosti 1 je rovna 0.33 ($\frac{1}{3}$). Podobně bychom mohli vypočítat změnu v okolí pro ostatní uzly.

Pro výpočet ChC se je však třeba vypočítat změnu ve všech vzdálenostech. Stačí nám pouze lehce upravit již nadefinovaný vztah. Změnu v okolí velikosti n, tedy definujeme následovně:

$$r^d_{t_1,t_2}(u) = \frac{|N^d_{t_1}(u) \Delta N^d_{t_2}(u)|}{|N^d_{t_1}(u) \cup N^d_{t_2}(u)|},$$

kde $N^n_t(u) = \{v \in V : d_t(u,v) = n\}$, tedy to jsou uzly spojené s uzlem u, v čase t, ve vzdálenosti n.

Nyní již známe vše, co potřebujeme pro výpočet ChC. Tu vypočítáme tak, že sečteme všechny změny ve všech vzdálenostech a pro výsledek, který můžeme porovnat s ostatními centralitami, normujeme. Vztah tedy bude vypadat následovně:

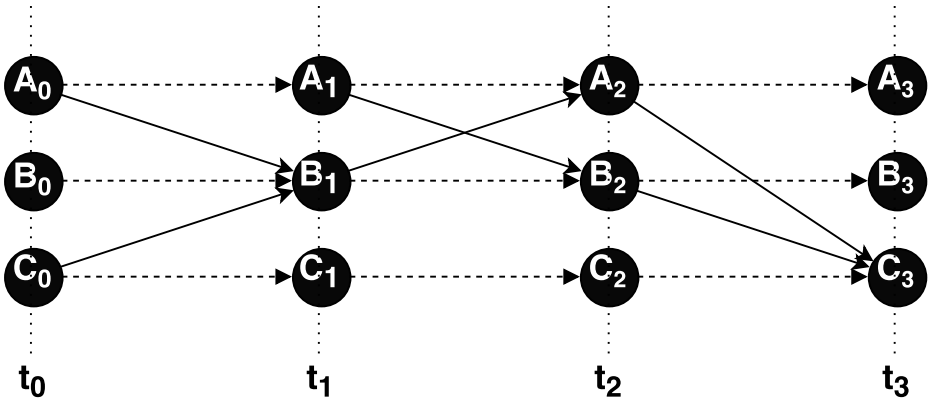
$$ChC_{t_1,t_2}(u) = \frac{1}{2} \sum_{n=0}^{|e_u|} \frac{1}{2^n} r^n_{t_1,t_2}(u),$$

kde $|\, e_u \,|$ je délka nejdelší cesty z uzlu u, do libovolného uzlu.

5 Algoritmy

V této kapitole se podíváme na jednotlivé algoritmy této práce. Pro každý z algoritmů v temporálních sítích si ukážeme dvě varianty výpočtu. Jednu pro průchod zepředu a druhou pro průchod zezadu.

Pro ukázku budeme průběh jednotlivých algoritmů demonstrovat na následujícím příkladu.



Obrázek 11: Příklad grafu pro algoritmy

5.1 Temporální nejkratší cesta (TNC)

Na vstupu tohoto algoritmu máme temporální graf $G(V, E)$ s okny P. Výstupem mají být všechny TNC ze všech uzlů do všech uzlů. Pro reprezentaci výsledků je nejvýhodnější zvolit matici $n \times n$, ve které hodnota na pozici $TNC_{u,v}$ reprezentuje délku TNC z vrcholu u do vrcholu v.

5.1.1 Průchod zepředu

V této variantě očekáváme postupně okna od nejstaršího po nejnovější. Tedy okna s hranami, u kterých je čas t nejnižší po okna s hrami s časem t nejvyšším.

Algoritmus:

Jako první vytvoříme matici TNC $n \times n$ a nastavíme všechny její hodnoty na Inf. Tzn., že všechny vzdálenosti jsou nekonečno, čili cesta mezi dvěma vrcholy neexistuje. Pro lehčí implementaci bude Inf i na diagonále, ačkoliv bychom zde mohli očekávat jiné číslo, třeba 0.

V dalším kroku postupně bereme jednotlivá okna a procházíme jejich hrany. Pokud je hodnota prvku $TNC_{u,v} > t$, kde t je n-té procházené okno, a v tomto okně existuje hrana

z u do v, pak za $TNC_{u,v}$ dosadíme hodnotu t. Pro náš příklad bude matice vypadat takto:

$$TNC = \begin{pmatrix} Inf & 1 & 3 \\ 2 & Inf & 3 \\ Inf & 1 & Inf \end{pmatrix}.$$

To ovšem nestačí. Musíme si uvědomit, že pokud se v okně t dostaneme z vrcholu u do vrcholu v a v okně $t+1$ z vrcholu v do vrcholu w, pak se také v okně $t+1$ dostaneme z uzlu u do uzlu w. Tedy stávající algoritmus při zpracování nové hrany musíme rozšířit tak, že pro každou novou hranu z vrcholu u do vrcholu v projdeme všechny pozice $TNC_{w,u}$, kde $w = 1, ..., n$, a pokud je hodnota $TNC_{w,u}$ menší než hodnota aktuálního okna a pokud je hodnota na pozici $TNC_{w,v}$ větší než hodnota aktuálního okna, pak nahradíme tuto hodnotu hodnotou aktuálního okna.

V našem případě vidíme, že existuje cesta mezi uzly C a A, ale v matici se na dané pozici vyskytuje hodnota Inf. Po úpravě algoritmu s přidáním kontroly bude již výsledek správný a bude následovně:

$$TNC = \begin{pmatrix} Inf & 1 & 3 \\ 2 & Inf & 3 \\ 2 & 1 & Inf \end{pmatrix}.$$

Algoritmus

VSTUP: Temporální graf $G(V, E)$ s okny $G_0, ..., G_P$.

VÝSTUP: Temporální nejkratší cesty mezi všemi vrcholy.

PSEUDOKÓD:

```
TNC[n][n] = Inf
foreach G_t in G
    foreach E_u,v in G_t
        if TNC[u][v] > t then
            TNC[u][v] = t
        foreach V_w in V
            if TNC[w][u] < t then
                if TNC[w][v] > t then
                    TNC[w][v] = t
```

ČASOVÁ SLOŽITOST: $P * m_t * n = m * n$

PAMĚŤOVÁ SLOŽITOST: n^2

5.1.2 Průchod zezadu

Jediný rozdíl na vstupu a výstupu u tohoto algoritmu bude v tom, že okna budeme procházet zezadu. Tzn., že jako první budeme zpracovávat hrany s vyšším časovým razítkem a až následně hrany s nižším časovým razítkem.

Algoritmus:

První krok tohoto algoritmu je podobný, avšak nenahrazujeme prvek hodnotou t, ale hodnotou 1. Tedy pro každou hranu, které vede z uzlu u do uzlu v nastavím pole $TNC_{u,v}$ na hodnotu 1. Tedy pro náš příklad vypadá matice takto:

$$TNC = \begin{pmatrix} Inf & 1 & 1 \\ 1 & Inf & 1 \\ Inf & 1 & Inf \end{pmatrix}.$$

Jak vidíme v předchozí matici, pokud bych prováděl pouze tento krok, nakonec bych měl v matici pouze hodnoty 1 na místech, kde někdy byla hrana. Proto, pokud přecházím z okna $t+1$ do okna t, tak inkrementuji všechny hodnoty o 1 a matice pak bude vypadat následovně:

$$TNC = \begin{pmatrix} Inf & 1 & 3 \\ 2 & Inf & 3 \\ Inf & 1 & Inf \end{pmatrix}.$$

Stále to však ke kompletnímu algoritmu nestačí. I zde si je třeba uvědomit, podobně jako u předcházejícího příkladu, že pokud existuje cesta v čase t z uzlu u do uzlu v a v čase $t-1$ hrana z w do u, pak v čase $t-1$ existuje i cesta z uzlu w do uzlu v. Nyní zde nastává drobný problém. Pro tuto operaci se musí uchovávat stav nejkratších cest předchozího okna, abych si nepřepsal stávající hodnotu novou a pro další uzly tak starou zanedbal.

Tedy pokud přidáváme hranu v čase t z uzlu u do uzlu v projdeme všechny pozice $TNC_{v,w}$ v čase $t+1$ a pokud pozice v čase t $TNC_{u,w} > TNC_{v,w} + 1$ v čase $t+1$ pak $TNC_{u,w}$ nahradím $TNC_{v,w} + 1$.

V našem příkladě opět chybí hrana z uzlu C do uzlu A. Po použití tohoto mechanizmu už výsledek bude shodný jako v předchozím příkladě. Výsledek je ukázán v následující matici:

$$TNC = \begin{pmatrix} Inf & 1 & 3 \\ 2 & Inf & 3 \\ 2 & 1 & Inf \end{pmatrix}.$$

Algoritmus

VSTUP: Temporální graf $G(V, E)$ s okny $G_0, ..., G_P$.

VÝSTUP: Temporální nejkratší cesty všech vrcholů.

PSEUDOKÓD:

```
TNC_t[n][n] = Inf
TNC_t+1[n][n] = Inf
foreach G_t in G
    TNC_t+1[n][n] = TNC_t[n][n]
    foreach E_u,v in p_t
        TNC_t[u][v] = 1
        foreach V_w in V
            if TNC_t[u][w] > TNC_t+1[v][w] + 1 then
                TNC_t[u][w] = TNC_t+1[v][w] + 1
    foreach V_u in V
        foreach V_v in V
            TNC_t[u][v] ++
```

ČASOVÁ SLOŽITOST: $P * m_t * n + P * n^2 = m * n + P * n^2$

PAMĚŤOVÁ SLOŽITOST: $2n^2$

5.1.3 Porovnání algoritmů

- Při průchodu zepředu může algoritmus skončit dříve, pokud již našel všechny cesty a nemusí tak procházet všechny hrany.
- I při předčasném ukončení algoritmu pro průchod zepředu získáme část nejkratších cest a nepotřebujeme procházet všechny hrany pro získání správného výsledku. Při ukončení po průchodu oknem t získáme všechny nejkratší cesty délky t.
- Algoritmus pro průchod zezadu musí vždy projít všechny hrany a nikdy nemůže skončit dříve, správný výsledek získáváme až po zkontrolování poslední hrany.
- Při průchodu zepředu potřebujeme polovinu paměti oproti průchodu zezadu.

5.2 Temporální closeness centralita(TCC)

Výpočet TCC se provádí z temporálních nejkratších cest. Je tedy třeba vypočítat TNC ve všech časech. Musíme tedy upravit předchozí algoritmy tak, abychom mohli vypočítat všechny cesty.

5.2.1 Průchod zepředu

Úprava proběhne tak, že u každého okna musíme vytvořit novou matici TNC pro dané okno a na konci algoritmu vypočítat hodnotu TCC.

Algoritmus

VSTUP: Temporal graf $G(V, E)$ s okny $G_0, ..., G_P$.

VÝSTUP: TCC všech vrcholů.

PSEUDOKÓD:

```
C[n] = 0;
foreach G_t in G
    TNC_p[n][n] = Inf
    foreach TNC_r
        foreach E_u,v in p_t
            if TNC_r[u][v] > t then
                TNC_r[u][v] = t
            foreach V_w in V
                if TNC_r[w][u] < t then
                    if TNC_r[w][v] > t then
                        TNC_r[w][v] = t
foreach TNC_r
    foreach V_u in V
        foreach V_v in V
            C[i] += 1 / (TCN_p[u][v] * (n-1) * P)
```

ČASOVÁ SLOŽITOST: $P*(1+2+...+P)*m_t*n+P*n^2 = m*n*\frac{(1+P)*P}{2}+P*n^2$

PAMĚŤOVÁ SLOŽITOST: $P * n^2 + n$

5.2.2 Průchod zezadu

Pro průchod zezadu se nám algoritmus upraví minimálně. S každým novým oknem nám stačí pouze připočítat normalizovanou hodnotu TNC k TCC. Je to dáno tím, že když projdu okno t, tak již znám TNC v čase t a nemusím procházet všechna okna jako u průchodu zepředu.

Algoritmus

VSTUP: Temporální graf $G(V, E)$ s okny $G_0, ..., G_P$.
VÝSTUP: TCC všech vrcholů.
PSEUDOKÓD:

```
C[n] = 0
TNC_t[n][n] = Inf
TNC_t+1[n][n] = Inf
foreach G_t in G
    TNC_t+1[n][n] = TNC_t[n][n]
    foreach E_u,v in p_t
        TNC_t[u][v] = 1
        foreach V_w in V
            if TNC_t[u][w] > TNC_t+1[v][w] + 1 then
                TNC_t[u][w] = TNC_t+1[v][w] + 1
    foreach V_u in V
        foreach V_v in V
            TNC_t[u][v] ++
            C[u] += 1 / (TCN[u][v] * (n−1) * P)
```

ČASOVÁ SLOŽITOST: $P * m_t * n + P * n^2 = m * n + P * n^2$
PAMĚŤOVÁ SLOŽITOST: $2n^2 + n$

5.2.3 Porovnání algoritmů

- Oproti TNC se složitost a hlavně paměťová náročnost pro průchod zepředu výrazně zhoršila, kdežto pro průchod zezadu zůstala podobná a liší se minimálně.
- Pro průchod zezadu je algoritmus pro výpočet TCC rychlejší než pro průchod zepředu.
- Na paměťovou náročnost algoritmu s průchodem zezadu nemá vliv počet oken, oproti tomu pro průchod zepředu s rostoucím počtem oken je algoritmus náročnější.

5.3 Temporální betweenness centralita (TBC)

Výpočet TBC se provádí z nejkratších cest a uzlů, kterými projdeme. Existuje-li více cest stejné délky, a tato cesta je nejkratší, musíme počítat se všemi takovýmito cestami.

5.3.1 Průchod zepředu

Pro tento průchod se mi nepodařilo vymyslet žádný efektivní algoritmus, který by správně pracoval. I podle [13] je paměťová složitost $m^2 * n^2$. Což již pro 40 uzlů a 400 hran bychom potřebovali okolo 1GB paměti při použití datového typu Integeer. Navíc, jak si ukážeme v následující kapitole, Gephi má omezené možnosti pro paměť.

5.3.2 Průchod zezadu

Pro průchod zezadu je algoritmus velmi podobný algoritmu pro výpočet TCC. Je třeba pouze uchovávat hodnoty uzlů, kterými jsme prošli a kolikrát jsme jimi prošli. V algoritmu nám tedy pouze přibude proměnná, která bude uchovávat počty průchodů jednotlivými uzly v daných cestách a pokud budou 2 cesty shodné délky, pak se tyto hodnoty průchodů sečtou.

Algoritmus

VSTUP: Temporální graf $G(V, E)$ s okny $G_0, ..., G_P$.

VÝSTUP: TBC všech vrcholů.

PSEUDOKÓD:

```
B[n] = 0
Nodes_t[n][n][n] = 0, Nodes_t+1[n][n][n] = 0
TNC_t[n][n] = Inf , TNC_t+1[n][n] = Inf
foreach G_t in G
    TNC_t+1[n][n] = TNC_t[n][n]
    Nodes_t+1[n][n][n] = Nodes_t[n][n][n]
    foreach E_u,v in p_t
        TNC_t[u][v] = 1
        Nodes_t[u][v][n] = 0
        Nodes_t[u][v][u] = 1
        foreach V_w in V
            if TNC_t[u][w] > TNC_t+1[v][w] + 1 then
                TNC_t[u][w] = TNC_t+1[v][w] + 1
                Nodes_t[u][w][n] = Nodes_t+1[v][w][n]
                Nodes_t[u][w][u]++
            if TNC_t[u][w] == TNC_t+1[v][w] + 1 then
                Nodes_t[u][w][n] += Nodes_t+1[v][w][n]
                Nodes_t[u][w][u] ++
    foreach V_u in V
        foreach V_v in V
            TNC_t[u][v] ++
            foreach V_w in V
                if u != v and v != w and u != w then
                    TBC += Nodes_t[u][v][w]/Nodes_t[u][v][u]
```

ČASOVÁ SLOŽITOST: $P * m_t * n + P * n^3 = m * n + P * n^3$

PAMĚŤOVÁ SLOŽITOST: $2n^3 + 2n^2 + n$

5.3.3 Porovnání algoritmů

Pro průchod zepředu sice není algoritmus implementován, ale můžeme porovnávat s hodnotami ze článku [13].

- Při porovnání paměťové náročnosti je lepší průchod zezadu. Můžeme si to demonstrovat následující tabulkou.

Počet uzlů	Počet vrcholů	Paměť zepředu	Paměť zezadu
30	300	300MB	100kB
50	500	2500MB	500kB
100	1000	40GB	4MB
100	5000	1TB	4MB

- Při porovnání výpočetní náročnosti můžeme také říci, že algoritmus pro průchod zezadu je rychlejší oproti algoritmické složitosti v již zmíněném článku.

5.4 Change centralita (ChC)

Pro tento algoritmus si ukážeme pouze jeden průchod. Je pak jedno, jestli tento algoritmus provedeme zepředu či zezadu. Dá se použít na obě strany a výsledek bude shodný.

Jako první si musíme určit pro obě okna, v jaké vzdálenosti se nachází dané uzly. Tedy určit z vybraného uzlu, velikosti nejkratších cest do ostatních uzlů. To můžeme provést následujícím algoritmem.

Algoritmus

VSTUP: Matice sousednosti A grafu G(V,E), vrchol w, ze které chceme cesty počítat

VÝSTUP: nejkratší cesty z uzlu w

PSEUDOKÓD:

```
cesta[n] = 0
tmp = false
foreach u in V
 if A[w][u]
    cesta[u] = 1
    tmp = true

delka = 1;
while tmp
    tmp = false;
    foreach u in V
     if cesta[u] == delka
        foreach v in V
            if A[u][v]
               if cesta[v] == 0
                  cesta[v] = cesta + 1
                  tmp = true
    delka++;
```

Nejprve v tomto algoritmu nastavíme vzdálenost z uzlu w do všech uzlů, se kterými je spojen na 1. A poté procházíme pole a kontrolujeme vzdálenosti pro číslo i. Je-li uzel v této vzdálenosti i spojen s jiným uzlem, pak kontrolujeme, zda již existuje kratší cesta. Pokud ne, nastavíme tuto hodnotu na velikost i + 1. Pole cest procházíme do té chvíle, dokud se nám nestane, že jsme již žádnou novou cestu nenašli. V každém novém procházení inkrementujeme hodnotu i o 1.

Poté stačí spočítat, kolik existuje uzlů v daných vzdálenostech, kolik uzlů má stejnou vzdálenost v obou oknech a kolik má různou vzdálenost v obou oknech. Stačí nám tedy projít pole cest obou oken a kontrolovat zda jsou hodnoty totožné či nikoliv. Pokud jsou totožné přidáme cestu v dané vzdálenosti do stejných polí stejných cest. Pokud jsou různé přidáme do pole různých cest jednu hodnotu.

Nakonec nám stačí tyto pole různých a stejných cest projít a dosadit hodnoty do vzorce. Jelikož nám Gephi umožňuje více oken, tak tento algoritmus provedeme pro všechny dvojice oken w a w+1.

6 Gephi

Gephi [4] je nástroj vyvíjený v jazyce JAVA nad NetBeans platformou [6, 17] sloužící pro vizualizaci a zkoumání vlastností komplexních sítí. Nástroj je možno stáhnout na stránkách [9].

6.1 Instalace a nastavení (systém Linux Mint 17)

Nejprve je třeba nástroj stáhnout z již zmíněných stránek [9] a rozbalit do libovolné složky v počítači. Nástroj spustíme pomocí scriptu gephi ve složce bin. Nelze-li nástroj spustit, bude nejspíše problém s verzí JDK ve vašem počítači. Nástroj Gephi je vyvíjen v JDK 1.7. a s novým JDK 1.8. zde nastaly problémy se zabezpečením a modifikátory. Pro správný běh je tedy vhodné využít starší verzi JDK. To můžeme provést změnou proměnné jdkhome v souboru etc/gephi.conf. Nemáte-li starší verzi JDK ve svém počítači, můžete si jí stáhnout na těchto stránkách [16]. Po provedení těchto změn by se vám měl nástroj již normálně spustit.

6.2 Gephi

Po spuštění Gephi a načtení libovolné sítě vidíme okno jako je na obrázku 12.

V horním panelu máme 3 možnosti. **Přehled**, to je aktuální okno, které vidíme. **Laboratoř dat** je tabulka, ve které vidíme všechna data o dané síti a vypočtené charakteristiky. Dále zde můžeme modifikovat síť a importovat či exportovat data. Poslední nabídka **náhled** mám slouží k vizualizaci dané sítě.

V levém panelu nastavujeme vzhled dané sítě, velikost jednotlivých uzlů a hran a můžeme zde také spustit různé algoritmy pro vizualizaci sítě.

V pravém panelu pak vidíme základní statistiku o dané síti a můžeme zde spouštět výpočet různých vlastností. Zde také najdeme vytvořený plugin pro výpočet temporálních vlastností.

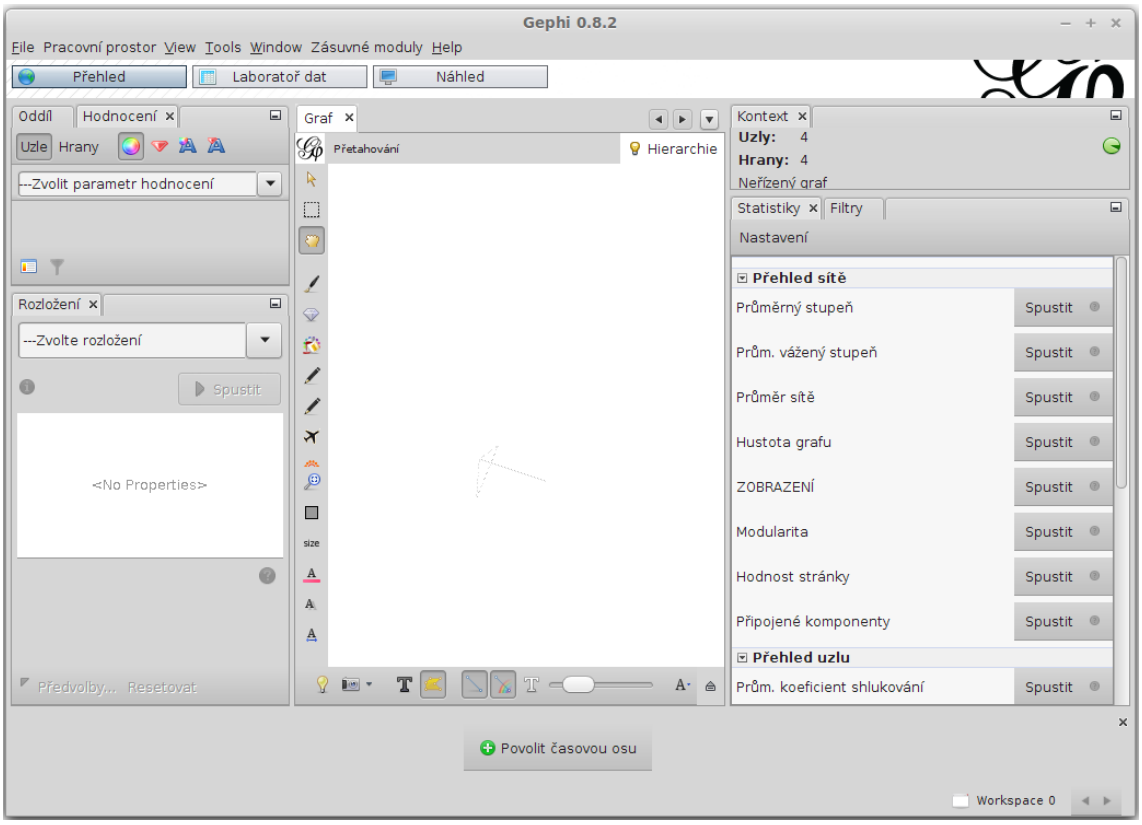
Velice zajímavý je pro temporální sítě dolní panel. Zde doporučuji vždy povolit časovou osu a v levé nabídce nastavit formát času na datum. Tím budeme moci počítat vlastnosti a vizualizovat síť pomocí jednotek pro měření času a nebudeme muset používat pouze čísla (Unixová časová razítka), které nám nic neřeknou.

Uvedl jsem zde pouze velice stručný a základní popis. Grafické uživatelské rozhraní je velmi intuitivní. Pro více informací o práci s Gephi doporučuji [19].

6.3 Přidání pluginů

Nyní si popíšeme jak do Gephi přidat námi vytvořené pluginy. V nabídce **Nástroje** klikneme na možnost **Zásuvné moduly**. Objeví se nám okno viz obrázek 13. V tomto okně zvolíme záložku **Stažené**.

Dále pak klikneme na tlačítko **Přidat zásuvné moduly** a vybereme všechny moduly z přiloženého CD. Ujistíme se, že jsou všechny moduly vybrány a klikneme na tlačítko Instalovat, obrázek 14. Dále již pokračujeme podle instalátoru v Gephi.



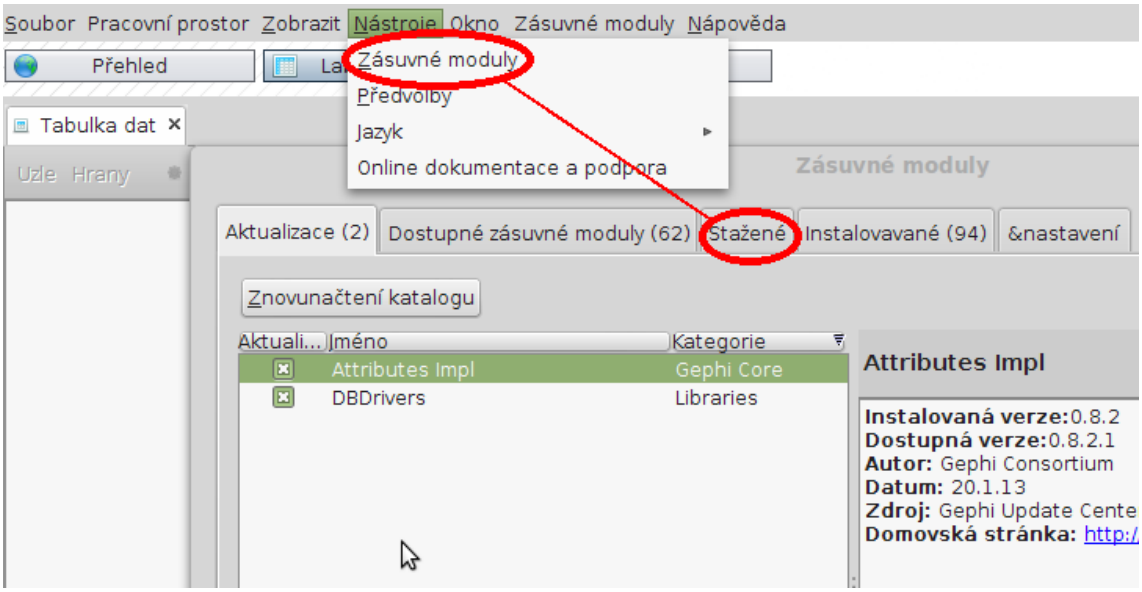
Obrázek 12: Okno Gephi

Může se stát, že se po nás budou chtít další zásuvné moduly. Tyto moduly musíme nainstalovat pro správný běh Gephi.

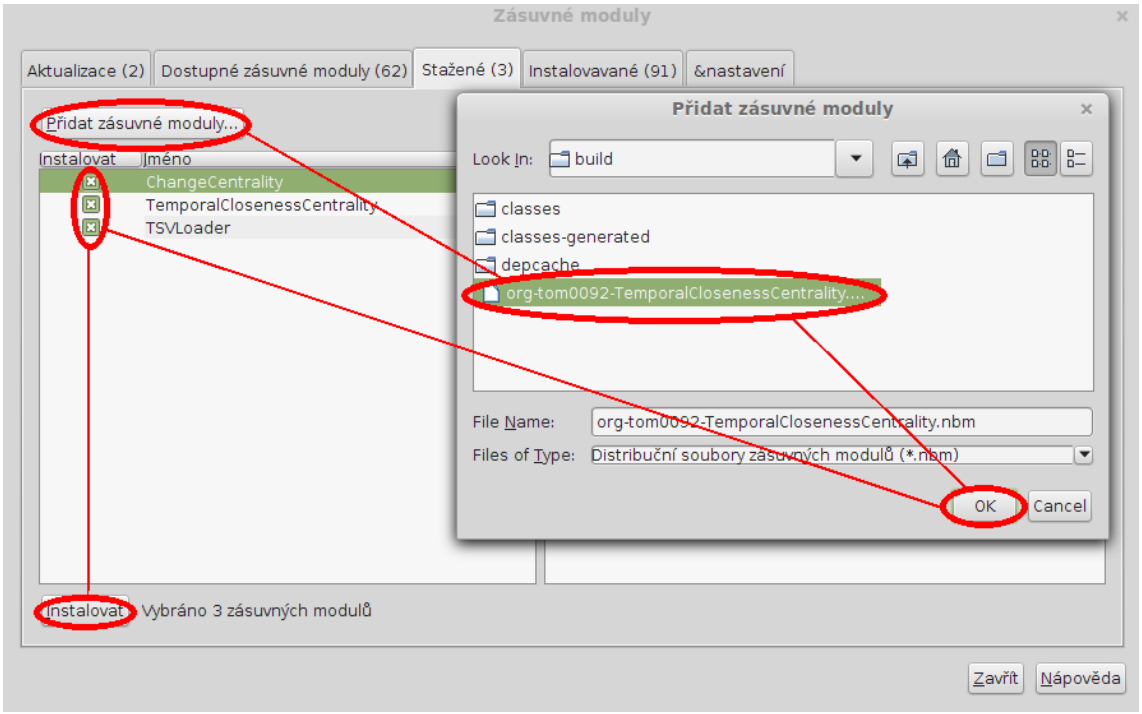
6.4 Vstupy pro Gephi

Nástroj Gephi nám dovoluje používat mnoho formátů. Pro dynamické sítě s časovým razítkem se však dá využít pouze formátu .gexf. Tento formát je však pouze čistý .xml soubor a je tedy pro větší sítě velmi velký. Např. pro datovou kolekci ENRON je velikost tohoto souboru okolo 150 MB. Vytvořil se proto plugin, který dokáže načíst soubor .tsv, který je znatelně menší. Pro ještě větší úsporu prostoru se tento soubor zkomprimuje a načítá pomocí streamů jazyka JAVA. Po této komprimací a využitím souborů .tsv se velikost vstupních souborů pohybuje v řádech jednotek kB, což je velmi vysoká úspora místa.

Soubor typu .tsv obsahuje nejprve 2 řádky s informacemi o síti a poté následují data sítě samotné. Na prvním řádku je informace o typu sítě. Zda se jedná o orientovanou či neorientovanou síť. Na druhém řádku se pak nachází informace o počtu uzlů a hran. Data jsou pak tvořena informacemi o všech hranách. Každá hrana je uvedena na jednom



Obrázek 13: Instalace krok 1



Obrázek 14: Instalace krok 2

řádku a dává nám informace o tom, ze kterého a do kterého uzlu daná hrana vede, její váhu a časové razítko, ve kterém hrana existuje.

Gephi celou síť ukládá objektově a jednotlivé objekty nejsou indexovány. Pro vložení nového objektu se tak musí projít všechny již vytvořené objekty a tak načítání větších sítí může být pomalejší. Z omezení, které bylo popsáno výše se tento problém neřešil, jelikož trvá-li načtení sítě dlouho, nebudeme moci na dané síti ani vypočítat temporální vlastnosti. Takže zde platí pravidlo, síť kterou se nepodaří načíst, nebo načítání trvá velice dlouho, tak na dané síti není možné vypočítat ani temporální vlastnosti.

6.5 Výstupy z Gephi

Výstupy jakékoliv vlastnosti vypočtené pomocí Gephi je buď soubor .html nebo již zmiňovaná tabulka dat. Výhodou výstupu .html je, že vypočtené charakteristiky můžeme uložit a poté opětovně prohlížet v libovolném prohlížeči.

Pro tvorbu grafů využívá Gephi balíčku JFreeChart, který můžeme stáhnout zde [12]. Ukázky některých grafů pak můžeme vidět zde [20, 10].

6.6 Tvorba pluginu v nástroji Gephi

Podrobné návody jak vytvářet nová menu, statistiky a jiné pluginy můžeme najít na wikipedii pro Gephi [22]. Nebude zde proto popis vytváření pluginů uvádět.

6.7 Omezení

Gephi je nastaveno na maximální využití paměti okolo 950 MB. Pokud je tato velikost překročena práce nástroje je zastavena. Nelze tedy zpracovávat příliš velké sítě a výpočet temporálních vlastností, které jsou velmi náročné, je tak značně omezen.

Další celkem nepříjemným problémem Gephi, je tvoření ID. Pokud načítáme novou síť pomocí našeho pluginu, používáme ID, které je nám dáno a je shodné s ID v záložce Laboratoř dat. Při zpracování však Gephi dává úplně jiné indexy daným uzlům než je zde uvedeno a pokud se pokusíme pomocí metody získat rozsah a počáteční ID, je nám vrácena hodnota uvedená právě v záložce Laboratoř dat a ne pozměněná hodnota ID. Abychom mohli tedy načíst novou síť, je třeba Gephi vypnout a znovu zapnout. Tuto chybu by bylo možno odstranit, avšak bychom museli vždy dvakrát procházet všechny hrany, což by bylo značně pomalé, nebo bychom si celou síť museli uchovat v paměti, což by byl značný problém a ubrali bychom si tak již malý prostor pro síť. Nejrychlejší a zároveň nejefektivnější je tedy nástroj restartovat, pokud chceme analyzovat jinou síť. Většina práce však spočívá v analýze 1 sítě, tedy tuto operaci nebudeme muset dělat tak často.

7 Vlastní nástroj

Jelikož jsou s Gephi již zmíněné problémy, které zde vznikly tím, že byl nástroj dodatečně rozšířen o možnost analýzy dynamických sítí a nebyl tomu úplně přizpůsoben, byl vytvořen i vlastní nástroj pro výpočet temporálních centralit. Tento nástroj nám umožní vypočítat TCC a TBC rychleji a na mnohem větších sítích, jelikož využívá algoritmu pro průchod sítě zezadu a danou síť celou nenačítá, ale analyzuje ji přímo ze souboru .gz. Jediné co musíme provést navíc, je setřídit daný vstupní soubor, což v porovnání s výpočtem TBC je zanedbatelné.

7.1 Vstupy

Pro výpočet temporálních centralit je zapotřebí .tsv souboru, který má seřazené hrany od nejstarší po nejnovější. Původní soubory však byly seřazeny náhodně či v opačném pořadí. V tomto nástroji tedy v první záložce najdeme možnost pro převod klasického .tsv souboru na potřebný .tsv soubor. Nástroj poté tento soubor načne, setřídí hrany, a uloží tak, jak potřebuje pro rychlý běh výpočtu. Výstupní soubor se bude jmenovat podobně. Bude mít navíc na začátku písmeno b (back). Tento soubor již pak můžeme využít k výpočtu temporálních centralit.

7.2 Výstupy

Výstupy algoritmu jsou hodnoty zapsané do souborů. Pro TCC se jedná o 2 soubory, první obsahuje hodnoty celkových TCC všech uzlů a druhý soubor obsahuje mezivýpočty TCC, stejně jako je tomu u Gephi. Pro TBB jsou výstupy pouze celkové TBC.

Tyto soubory lze v Linuxu zpracovat velmi snadno pomocí gnuplot. Stačí využít následující skript pro vykreslení křivky.

Script

```
set key off
set title "Temporalni Closeness (Betweenness) centralita" font ",20"
set xlabel "uzel"
set ylabel "TCC" (TBC)
plot './clodata.tsv' (betdata.tsv) using 1:2 with points pointsize 0.5
     pointtype 7
pause -1 "Hit any key to continue"
```

A následující pro zobrazení výsledků mezivýpočtu.

Script

```
set key off
set title "TCC␣mezivypocet:␣CC␣v␣okne" font ",20"
set xlabel "okno"
set ylabel "ID␣uzlu"
set palette defined (0 1 1 1, 1 0 0 0)
plot './cloAll.tsv' using 1:2:3 with image
pause -1 "Hit␣any␣key␣to␣continue"
```

8 Analýza vybraných datových kolekcí

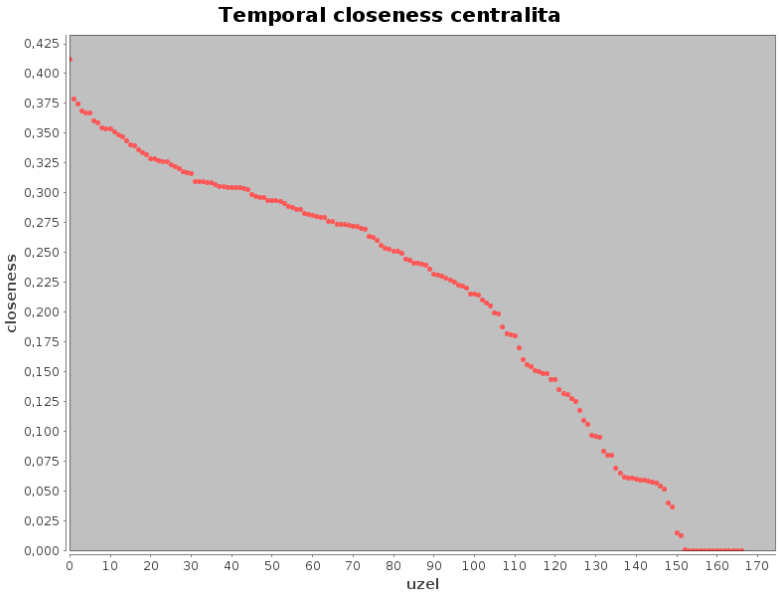
V této kapitole se podíváme na 3 různé datové kolekce, které zkusíme zanalyzovat pomocí již uvedených algoritmů. Všechny 3 kolekce nalezneme v příloze.

8.1 Emailová komunikace

První datová kolekce [3] je emailová komunikace zaměstnanců jedné firmy. Vrcholy nám reprezentují jednotlivé zaměstnance a hrany nám reprezentují poslaný email. Tato síť má 167 uzlů a okolo 83 000 hran. Data byla shromažďována prvních 9 měsíců roku 2010. Jelikož má email vždy odesílatele a příjemce, jedná se tedy o orientovanou síť.

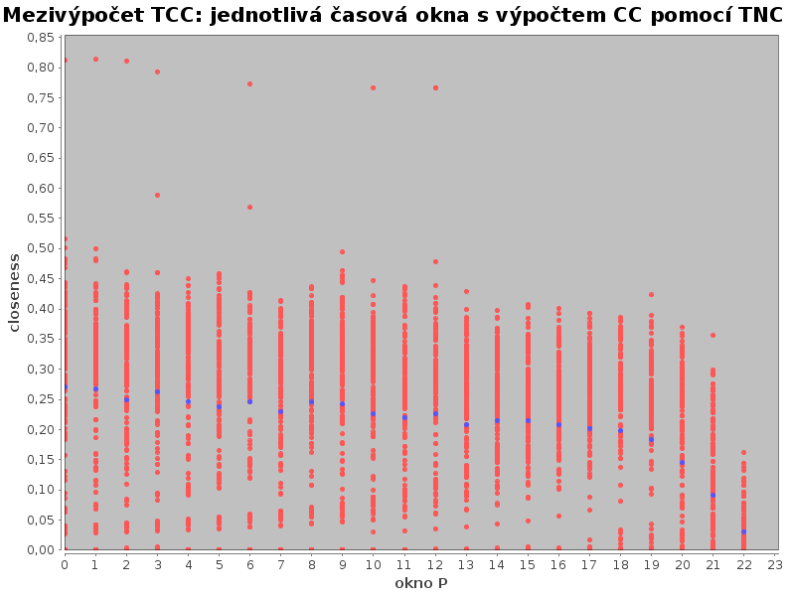
8.1.1 Temporální closeness centralita (TCC)

Jako první se podíváme na TCC. Na obrázku 15 můžeme vidět TCC pro všechny uzly s velikostí okna 7 dnů. Podíváme-li se blíže, vidíme, že v této síti existuje jeden uzel, který má značně vyšší TCC než ostatní uzly. Dále asi 100 uzlů má nadprůměrnou TCC a pak zde existují uzly, které mají téměř nulovou TCC. Nejspíše se bude jednat o jedince, kteří neměli skoro žádnou komunikaci a email téměř nepoužívali.



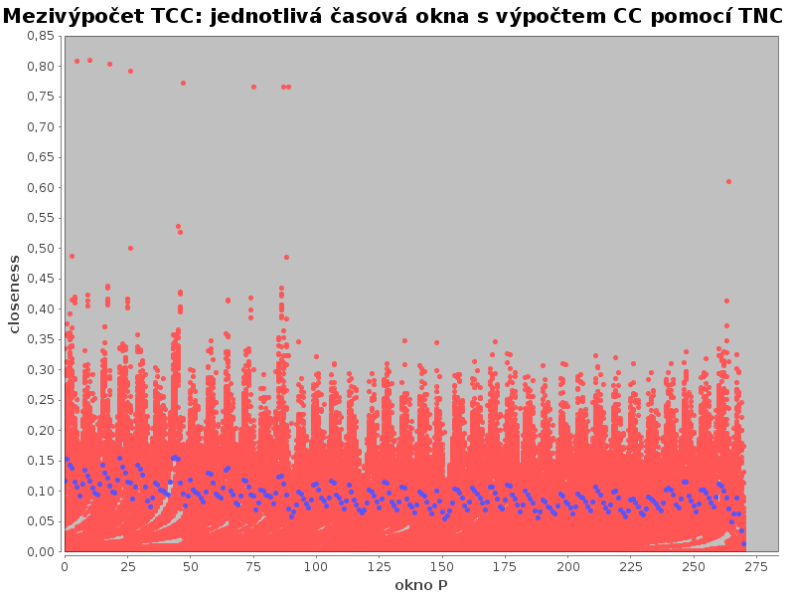
Obrázek 15: TCC - velikost okna = 7 dnů

Zajímavější než samotná TCC však může být mezivýpočet této hodnoty, který vidíme na obrázku 16. Jedná se o vypočítanou CC za pomocí TNC. V daném čase je vynesena hodnota CC všech uzlů. Modře je pak znázorněna průměrná hodnota. Jak vidíme, tato hodnota je přibližně stejná po celý interval jen ke konci klesá. To je způsobeno ukončením dat, a proto se v posledních oknech tato hodnota pomalu blíží 0.



Obrázek 16: Mezivýpočet TCC - velikost okna = 7 dnů

Zajímavější však je, že zde v některých oknech existují jedinci s velmi vysokou hodnotou oproti ostatním. Pokud bychom si zkusili zmenšit okno na velikost jednoho dne, takovéto extrémy by nám stále zůstaly. Můžeme to vidět na obrázku 17.

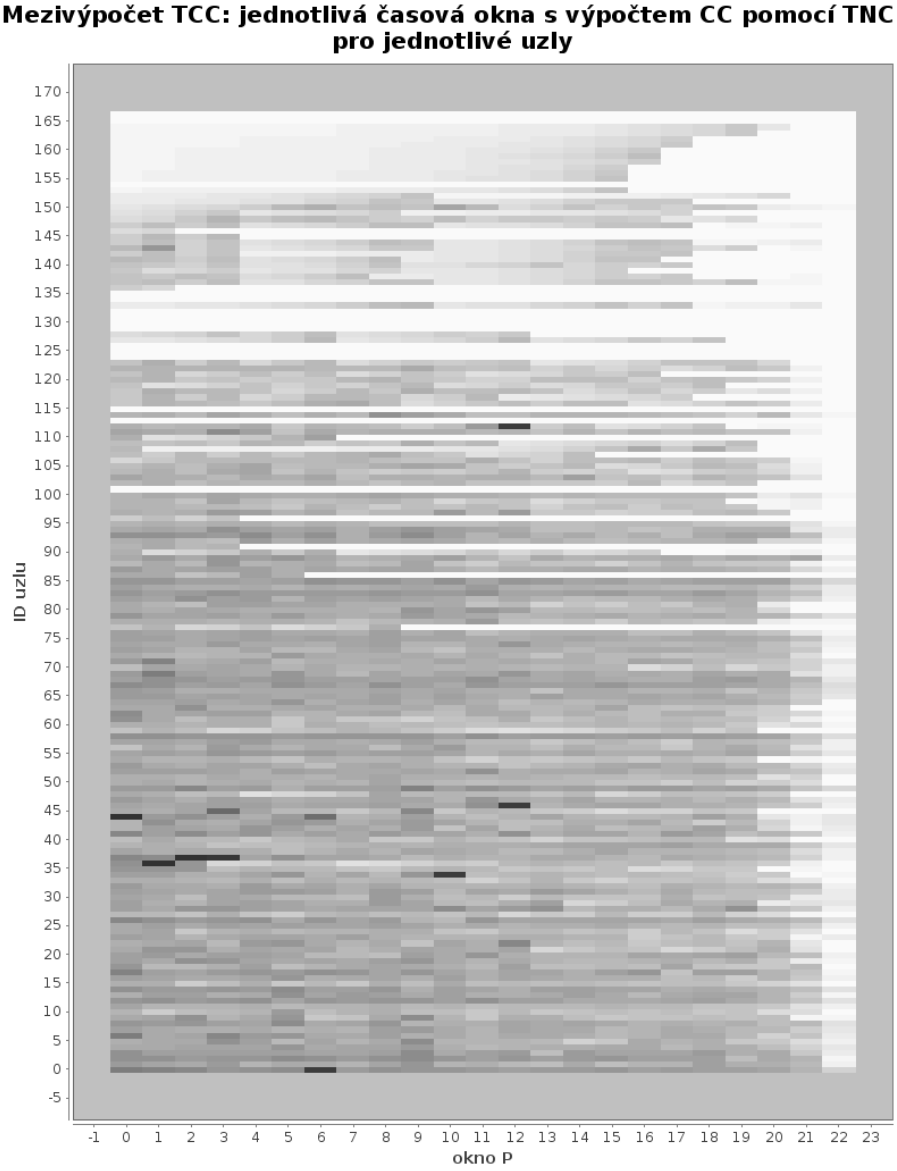


Obrázek 17: Mezivýpočet TCC - velikost okna = 1 den

K těmto jedincům se za chvíli vrátíme. Nejdříve se však ještě podíváme co můžeme

vyčíst z grafu s posunem 1 den. Vidíme, že nám průměrná hodnota začala oscilovat a i samotné hodnoty CC jsou s určitou periodicitou vyšší a nižší. Když se zamyslíme, toto bude nejspíše způsobeno víkendy. O víkendu pracuje méně lidí, možná také nikdo, takže je komunikace menší a tedy i CC je nižší. Přes týden pak lidé pracují, komunikace je větší a CC se tedy zvýší. Nalezli jsme nějakou periodicitu v této kolekci.

Nyní zpátky k jedincům, kteří měli v daných oknech vysokou hodnotu CC. Jistě by nás zajímalo, zda se jedná o jednoho jedince, či těchto vysokých hodnot dosahují různí jedinci. K tomu využijeme následujícího obrázku 18.



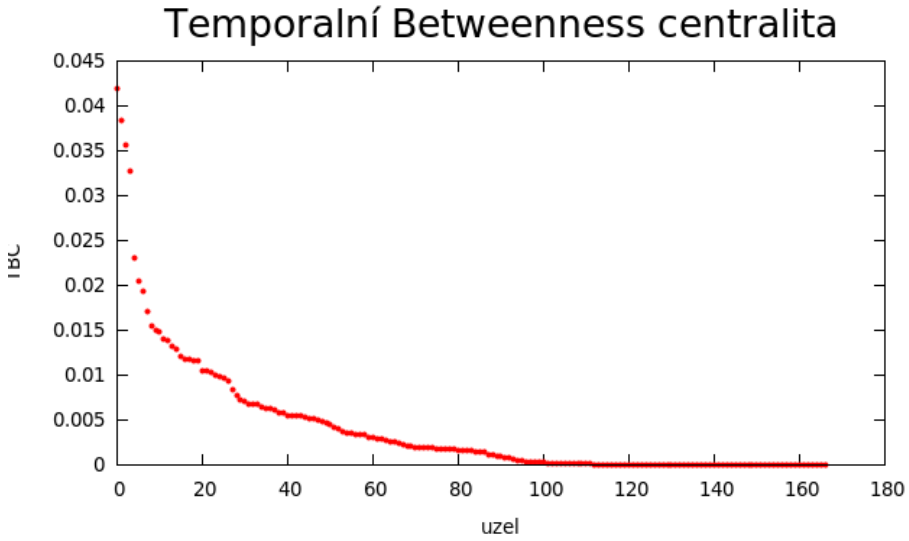
Obrázek 18: CC jednotlivých uzlů - velikost okna = 7 den

Jedná se o vizualizaci dat pomocí heatmap, kterou získáme mezivýpočtem jako předchozí vizualizaci, avšak nyní vidíme, který uzel má jakou CC. Je-li u uzlu bílá hodnota, znamená to, že CC je v tomto okně rovna 0. Čím je pak okýnko tmavší, tím je daná hodnota CC vyšší. Můžeme vidět, že dané extrémy jsou pro různé uzly a nejedná se tak o jeden jediný uzel. Další zajímavostí je, že pokud bychom si našli celkovou hodnotu TCC těchto uzlů, byla by tato hodnota u většiny průměrná či podprůměrná. Tedy se nejedná o uzly, které měly jednu z nejvyšších hodnot TCC.

Z posledního grafu také můžeme vyčíst, že zde existují opravdu uzly, které vůbec nekomunikují. Jedná se o uzly v horní polovině grafu. Vidíme, že ve všech oknech mají bílé políčko, tedy hodnota CC je 0, tedy nejspíše neexistovala žádná cesta. Dále také vidíme, že zde byly uzly, které přestaly v daném okně komunikovat a jejich CC je od jistého okamžiku nulová. Tedy komunikovali, ale od jistého okamžiku komunikace úplně vymizela a již nebyla tímto uzlem odeslána žádná zpráva.

8.1.2 Temporální betweenness centralita (TBC)

V dalším kroku se podíváme na výsledky TBC. Výsledný graf můžeme vidět na následujícím obrázku 19. Vidíme, že v dané síti existuje pár uzlů s vysokou TBC a skoro polovina uzlů s téměř nulovou TBC. Ostatní uzly mají hodnotu TBC mezi 0 - 0.3. Tento výpočet však probíhal na velikosti okna 30 dnů. Nedá se tedy srovnávat s výsledky TCC s velikostí okna 1 a 7 dnů a výpočet pro kratší okno nám způsobí extrémní růst TNC.

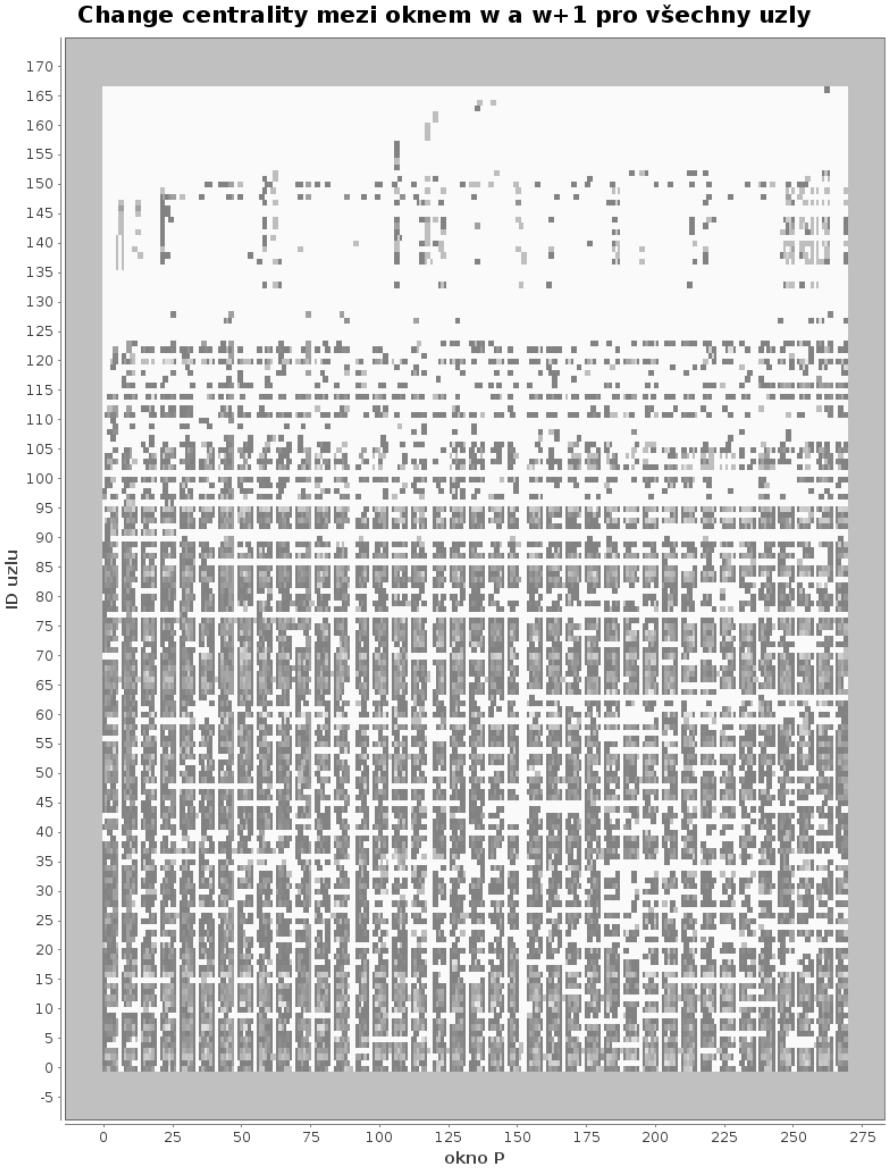


Obrázek 19: TBC jednotlivých uzlů - velikost okna = 30 dnů

8.1.3 Change centralita

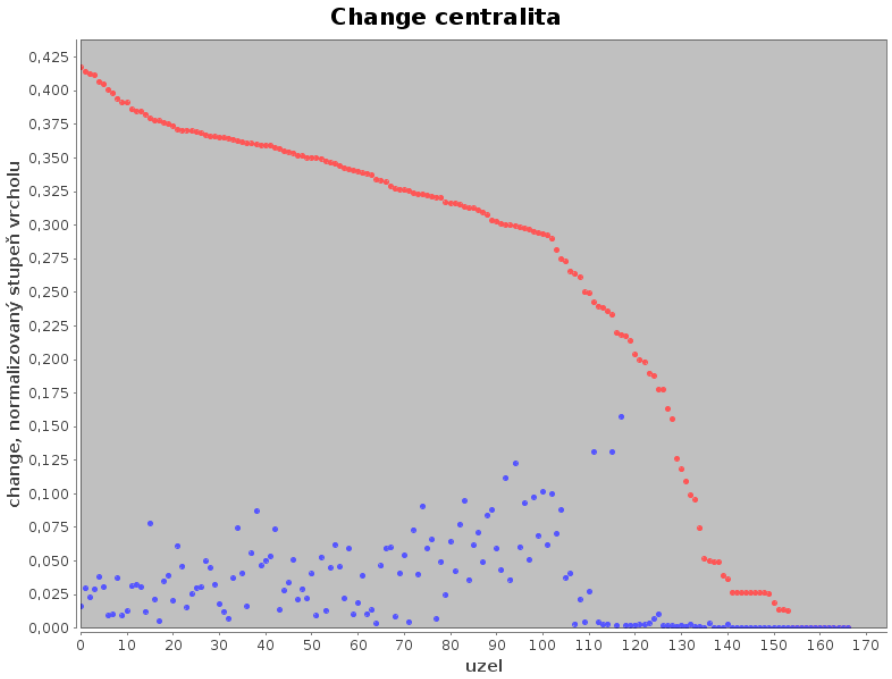
Jako poslední se podíváme na výsledky ChC. Když se podíváme na všechny změny všech uzlů, které můžeme vidět na následujícím obrázku 20, uvidíme, že změny probíhají opět

v cyklech. Pár dní se dějí změny, což je tmavší barva a poté vždy následuje pár dní, kdy se žádné změny nedějí. Což bude nejspíše způsobeno víkendy, jak již bylo řečeno dříve.



Obrázek 20: ChC jednotlivých uzlů - velikost okna = 1 den

Dále pak můžeme vidět, že uzly v horní části příliš velké změny nemají a tedy jsou stabilnější. Nejspíše to je způsobeno tím, že dané uzly nekomunikují a tedy se jejich okolí nemění. Můžeme si to ověřit na následujícím obrázku 21. Červeně je vyznačena velikost změn, tedy ChC a modře je normalizovaná hodnota stupně daného uzlu. Opravdu pak vidíme, že vrcholy s velmi malou a nulovou ChC mají i nízký stupeň uzlu.



Obrázek 21: ChC společně s normovanou Degree centralitou - velikost okna = 7 den

Zajímavé jsou pak 3 uzly mezi hodnotami 110 a 120. Přestože mají nejvyšší hodnotu stupně vrcholu, jejich hodnota ChC je poměrně nízká. Což značí, že tyto uzly komunikují často, ale nejspíše se stejnými uzly a v pravidelných intervalech. Jsou tedy stabilní a velmi aktivní. Na podrobnější informace o těchto uzlech bychom se museli podívat do Gephi do záložky Laboratoř dat.

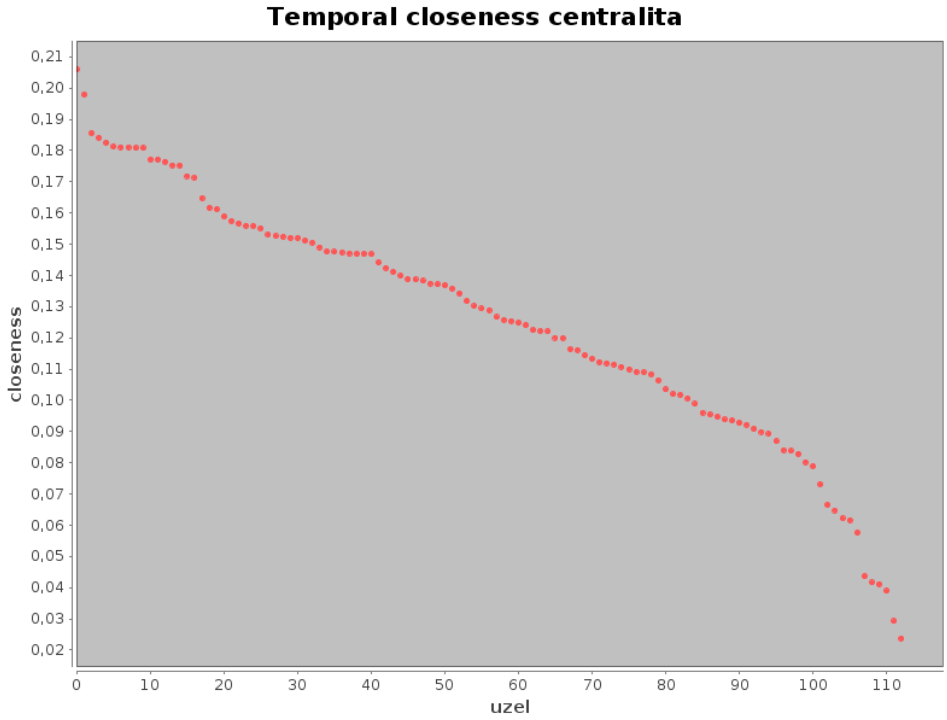
Zanalyzovali jsme si tedy první datovou kolekci. Zjistili jsme, že se aktivity uzlů mění v týdenních cyklech. Existuje zde pár uzlů, které měly některé dny velmi vysokou komunikaci oproti normálnímu týdnu a tyto uzly jsou odlišné. Dále jsme se podívali na TBC, které nemá až tak vypovídající hodnotu, jelikož její výpočet je velmi obtížný a podařilo se nám ho provést pouze s velikostí okna 30 dnů. Dále se již touto centralitou nebudeme zabývat. Dále jsme porovnali hodnotu ChC s velikostí stupně vrcholu a nalezli 3 uzly, které byly velmi aktivní a byli také velmi stabilní.

8.2 Konference

Tato síť [2] nám reprezentuje, kdy spolu lidé navázali kontakt. Jedná se o data shromážděná na konferenci, které se účastnilo 119 návštěvníků, kteří nám tvoří uzly. Hrany mezi uzly znamenají, že v daný okamžik došlo ke společnému kontaktu tváří v tvář po dobu minimálně 20 sekund. Tato konference trvala tři dny a bylo zaznamenáno okolo 20 000 kontaktů. Oproti předchozí síti je tato síť neorientovaná.

8.2.1 Temporální closeness centralita (TCC)

Jako první se opět podíváme na TCC, kterou můžeme vidět na následujícím obrázku 22. Oproti prvnímu příkladu nám TCC klesá pomaleji a až na pár posledních a prvních uzlů tvoří téměř přímku. Lze tedy předpokládat, že v této síti byly aktivní skoro všechny uzly a bude zde minimum uzlů, které by měly minimální kontakt s ostatními.

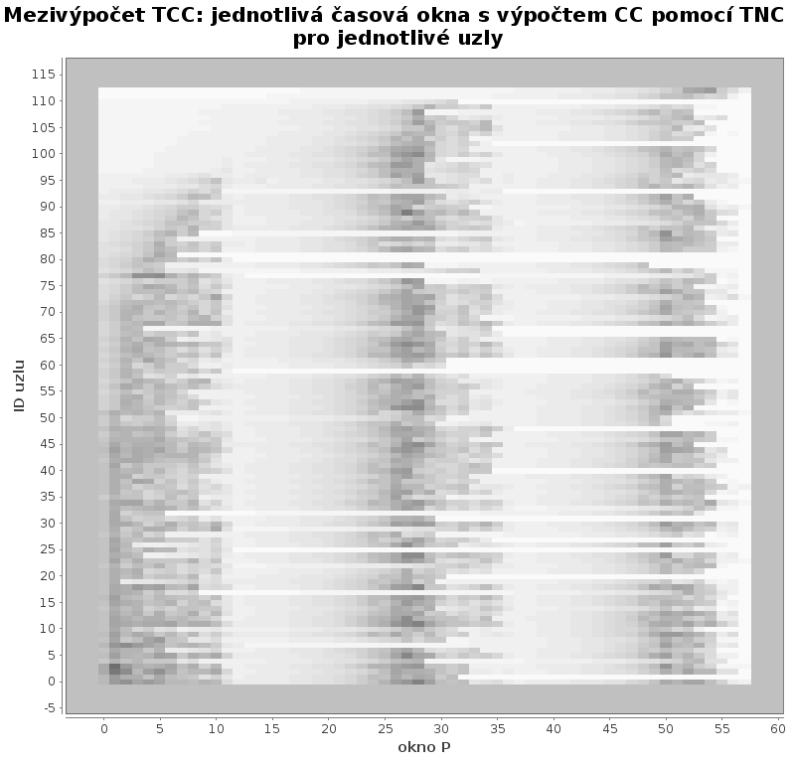


Obrázek 22: TCC - velikost okna = 1 hodina

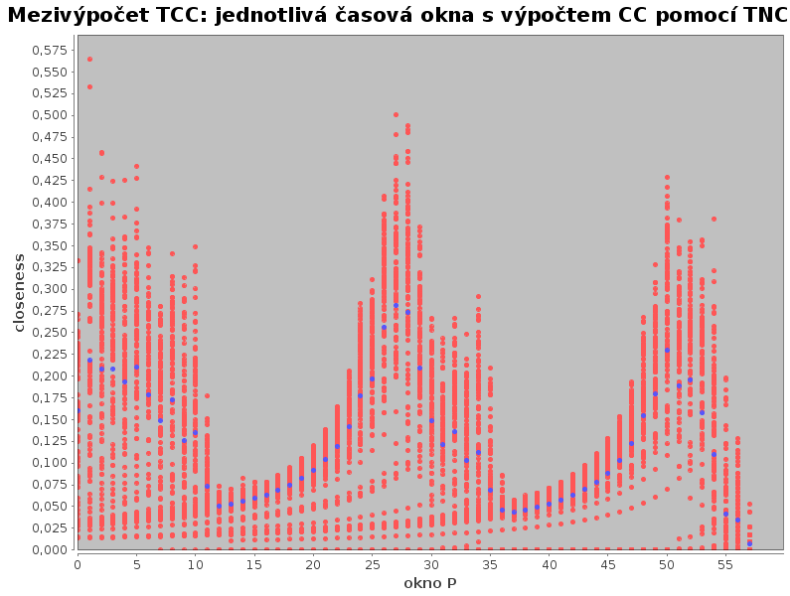
Vidíme zde 2 uzly, které mají o něco vyšší TCC než ostatní a 6 uzlů, které měli nižší TCC než ostatní. Lze tedy předpokládat, že uzly s nižší TCC se konference účastnily pouze 1 den. Naší domněnku si můžeme ověřit na následujícím obrázku 23.

Opravdu můžeme vidět, že zde existují lidé, kteří se konference zúčastnili pouze jeden den. Tyto uzly, reprezentující lidi, pak již mají v daném řádku pouze bílý pruh, což znamená, že již nebyly aktivní a jejich CC byla rovna 0. To znamená, že neexistovala žádná cesta mezi nimi a ostatními uzly. Z tohoto grafu můžeme také vidět, že se CC periodicky zvyšuje a snižuje. Je to dáno změnou den a noc. Dále vidíme, že konference se konala opravdu 3 dny, což je dáno třemi tmavšími svislými pruhy v grafu, které jsou z doby konference a dále mezi nimi můžeme vidět 2 bílé pruhy, které značí noc.

Periodicitu si také můžeme ověřit na posledním obrázku 24.



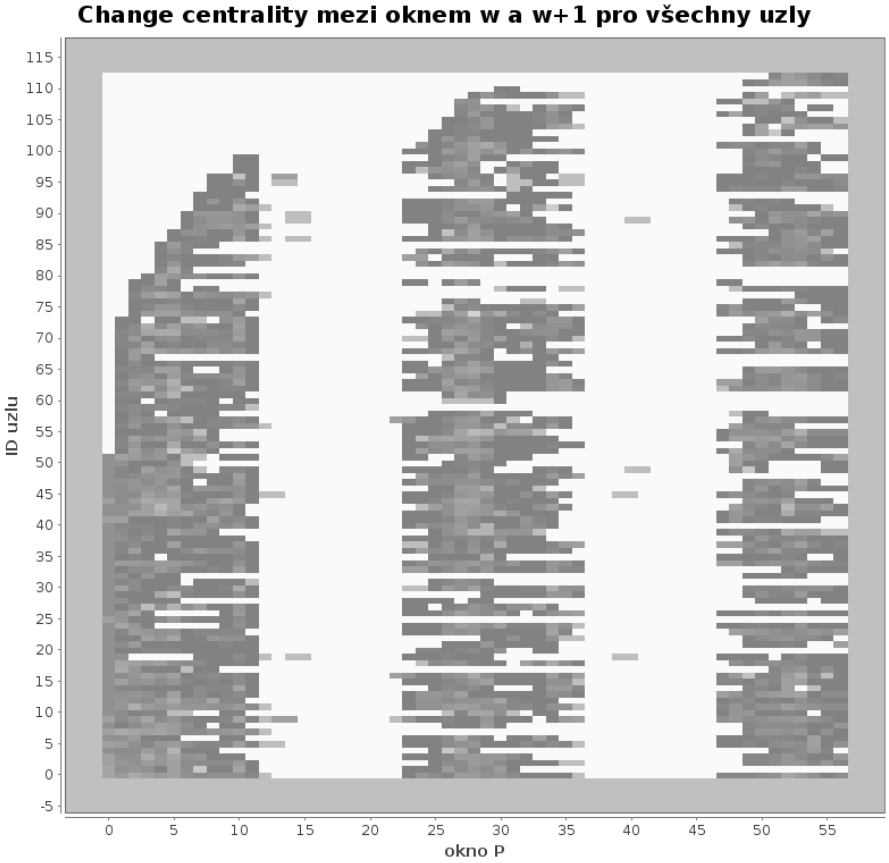
Obrázek 23: mezivýpočet TCC - velikost okna = 1 hodina



Obrázek 24: mezivýpočet TCC - velikost okna = 1 hodina

8.2.2 Change centralita

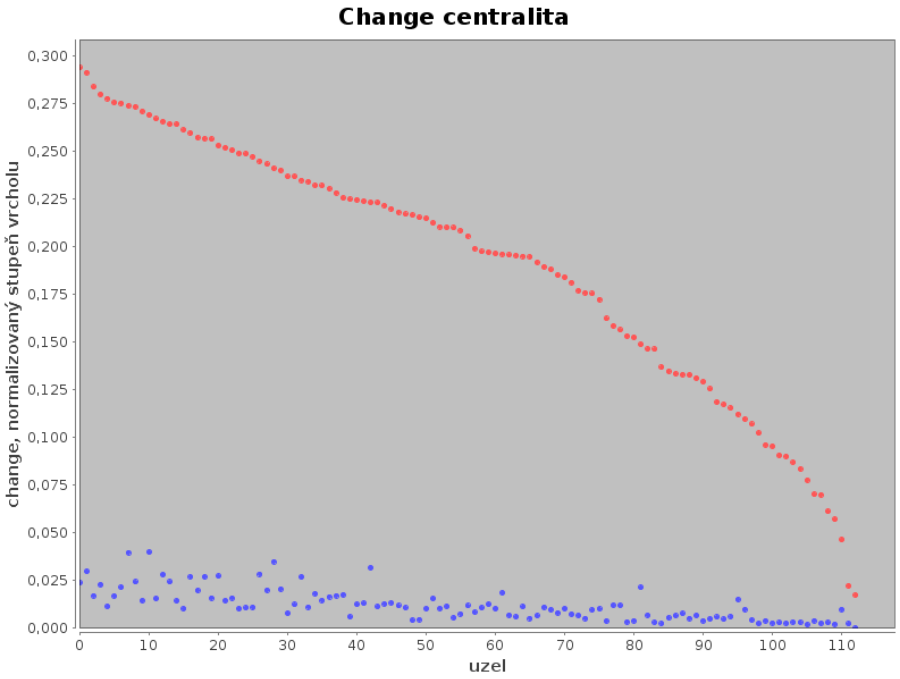
Nyní se podíváme na ChC, obrázek 25. Zde si můžeme ověřit, že zde byli lidé, kteří na konferenci byli pouze 1 den a pak zde již nebyli. Dále nám zřetelněji vylezli i další lidé, kteří se konference zúčastnili až později. Chyběli část prvního dne nebo celý první den. Můžeme předpokládat, že se jedná o uzly s nižší ChC a také nižší CC.



Obrázek 25: ChC jednotlivých uzlů - velikost okna = 1 hodina

Další ChC centralitu můžeme vidět na následujícím obrázku 26. Na tomto grafu můžeme vidět ChC porovnanou se stupněm vrcholu. Oproti první síti, zde nejsou podobné uzly, které by měly vysoký stupeň a zároveň nízkou ChC. V tomto grafu uzly s vyšším stupněm mají spíše vyšší ChC a uzly s nižším stupněm mají spíše nižší ChC.

Zanalyzovali jsme si i druhou síť, která oproti první byla neorientovaná. I zde jsme objevili periodu, nyní velikosti jednoho dne. Dále jsme porovnali vývoj TCC, kde můžeme říci, že u této sítě měli více uzlů nadprůměrnou TCC, což je zapříčiněno, že v této síti bylo více uzlů aktivních a bylo zde pouze pár lidí, kteří se konference zúčastnili pouze jeden den. Hodnoty ChC také klesaly pomaleji a byly více úměrné stupni. Nebyli zde nalezeni lidé s vysokým stupněm a nízkou ChC.



Obrázek 26: ChC společně s normovanou Degree centralitou - velikost okna = 1 hodina

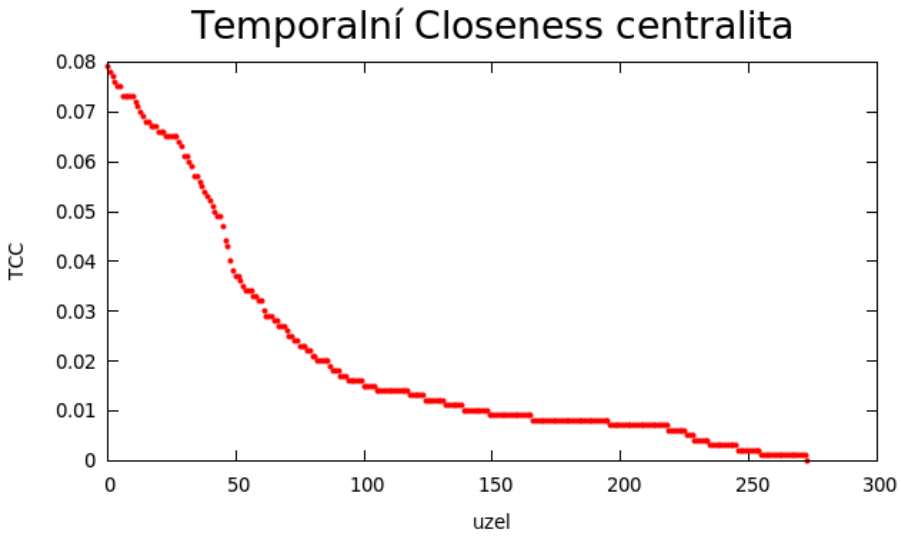
8.3 Lidský kontakt

Poslední datová kolekce [1] je podobná předchozí, byly zde však zaznamenávány údaje od 274 lidí po dobu 4 dnů. Celkem bylo získáno okolo 28 000 hran. U této kolekce si pouze demonstrujeme funkčnost našeho řešení TCC.

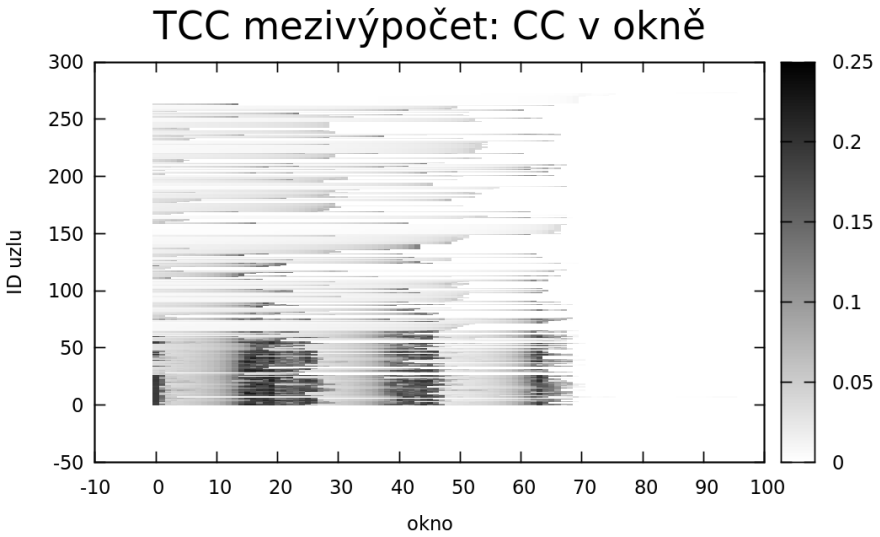
8.3.1 Temporální closeness centralita (TCC)

Jako první se opět podíváme na výsledek celkových TCC. Jak vidíme na obrázku 27, výsledná křivka se liší od křivky z minulé datové kolekce. Nejspíše to znamená, že se daní lidé neviděli pravidelně a v tak hojném počtu jako v předchozím případě. Existují zde uzly, které mají velmi vysokou TCC, tedy mají ke všem blízko. Dále pak jsou uzly s vysokou TCC, avšak není jich mnoho, okolo 50. Zbylé uzly pak mají podprůměrnou hodnotu TCC. Pokud bychom tento graf porovnali s grafem získaným pomocí Gephi, zjistili bychom, že jsou si tyto křivky podobné.

Podíváme se také na to, jak vypadají mezivýpočty TCC. To vidíme na dalším obrázku 28. Je tedy pravdou, že zde existuje pár uzlů, kteří jsou si v daných okamžicích velmi blízko. Tedy se s daným okolím často a pravidelně stýkají. Jedná se zhruba o 50 uzlů, které jsou v tomto grafu dole. Jejich CC hodnota se periodicky opakuje jako tomu bylo u předchozích dat. U ostatních uzlů se hodnota CC mění spíše náhodně, bez většího náznaku opakování. Tedy tito lidé se s ostatními stýkali spíše náhodně a méně často.



Obrázek 27: TCC - velikost okna = 1 hodina



Obrázek 28: mezivýpočet TCC - velikost okna = 1 hodina

Zanalyzovali jsme si tedy alespoň částečně 3 sítě a dostali jsem 3 různé křivky pro výslednou TCC. První nám nejprve pozvolna klesala, následně začala strměji skokově padat a nakonci se vyskytovala skupina uzlů s téměř nulovou TCC. Druhá křivka se blížila přímce, kde pouze pár začátečních uzlů mělo vyšší hodnotu TCC a pár koncových uzlů mělo nižší hodnotu TCC. Poslední křivka pak měla několik uzlů s vysokou TCC a hodně uzlů s nízkou TCC.

Ve všech datech se dala vyčíst určitá periodicita mezivýpočtu TCC. Ať již denní či týdenní. Tyto cykly jsou způsobeny režimem člověka a to střídání dne a noci a střídání

pracovních dnů s víkendy a volny. V těchto mezivýpočtech se pak dalo nalézt i občasné nestandartní chování některých uzlů, které by se dalo dále analyzovat.

TBC se bohužel dá vypočítat pouze na velkých oknech, kterých není mnoho. Tyto charakteristiky se spíše blíží statickým hodnotám a zde jsou uvedeny pouze jako příklad u první datové kolekce.

Pomocí ChC jsme pak ověřili naše úvahy. Za pomocí velikosti stupně daného uzlu jsme zjistili, že platí, že čím má daný uzel větší stupeň, tím má větší i změny okolí. Pouze u prvního příkladu jsme našli několik uzlů, které se tomuto pravidlu vymykaly. Tyto uzly by stály za důkladnější popis pomocí dalších vlastností.

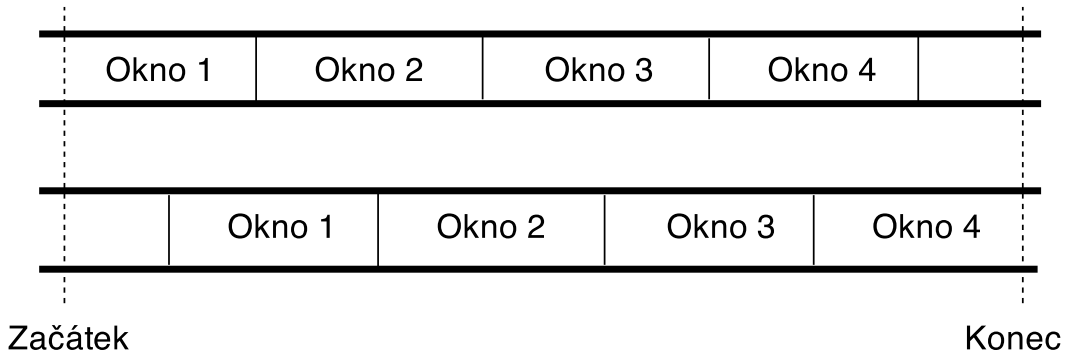
9 Rozbor vzniklých problémů

V této kapitole se podíváme na porovnání Gephi a vlastního nástroje. Projdeme si mezery a problémy, které nastaly a možné postupy jak tyto problémy řešit. Dále se zaměříme na možné rozdíly mezi výsledky a proč k nim může dojít.

9.1 Rozdíly ve výsledcích

Nastavili jsme stejný začátek i stejný konec a přesto se nám výsledky liší. Tohle se může stát a není to způsobeno chybou algoritmu samotného, ale tím jak daný algoritmus pracuje. Algoritmus s průchodem zepředu si nejprve vytvoří okno od počátku a poté se posouvá s velikostí okna. Může se stát, že tedy nedojde přesně ke konci. Implementačně je dáno zda se poslední okno zachová, zda se vynechá, či ho započítáme klasicky nebo si interval prodloužíme.

U průchodu zezadu vzniká stejný problém. Avšak na opačném konci. Celý tento proces můžeme vidět na následujícím obrázku 29.



Obrázek 29: Rozvržení oken

Nyní již vidíme v čem nastává problém. Při průchodu zepředu jsou data v daném čase zpracovávána v jiném okně než při průchodu zezadu. Avšak algoritmus je správný. Nastavíme-li počátek a konec tak, aby nám vyšla okna stejná, budou i výsledky stejné. Toto bylo testováno na příkladu z [13] a výsledky vyšly pokaždé stejně a totožné s výsledky v tomto článku.

9.2 Přetečení long

Jednou z nepředvídaných nástrah bylo při počítání počtu cest přetečení celočíselného typu long. Proč k tomu došlo? Vznikne-li v časově uspořádaném grafu nějaká smyčka, jako bylo například na obrázku 9, způsobí navýšení počtu cest na dvojnásobnou hodnotu. Každá další smyčka pak způsobí další nárůst počtu cest. Obecně můžeme říci, že

při n takových smyčkách se mi počet cest rovná 2^n. Stejným způsobem se pak navyšují průchody jednotlivými vrcholy. Pokud vedly cesty stejnými vrcholy, budou se také tyto průchody násobit. Jelikož je velikost největšího datového typu pro jazyk Java rovna 2^{64} již po 64 takových smyčkách mi tento typ přeteče a nastává problém. A jestliže má graf 2800 hran mezi 200 vrcholy, dá se předpokládat, že takové smyčky se zde budou objevovat a celkem často. Tedy je třeba vyřešit tento problém. Počítání smyček nám však nebude stačit. Průchody daným uzlem se tak jednoduše počítat nedají. Mám-li dvě cesty stejné délky, kdy jedna vede přes uzel A jedenkrát, a má 1 smyčku, a druhá druhá vede také přes uzel A ale dvakrát, a má 2 smyčky, musím tyto cesty sečíst. Tedy počet výsledný počet cest přes tento uzel bude $1 * 2 + 2 * 4 = 10$. Pokud bychom pouze sčítali smyčky a průchody, dostali bychom výsledek $3 * 8 = 24$, což je špatně. Je tedy třeba navrhnout jiné řešení.

9.3 Srovnání délky běhu

Porovnávali jsme složitosti daných algoritmů. Nyní se podíváme na délku běhu již naimplementovaných algoritmů. V první tabulce 3 vidíme časy běhu vlastní aplikace pro daný algoritmus. Máme zde délku běhu převodu a seřazení všech hran, poté délku běhu TCC pro velikost okna rovnou jedné hodině a pro emailovou komunikaci i s velikostí okna 1 den.

	Převod	TCC - 1 hodina	TCC - 1 den
Emailová komunikace	0.75 s	8.3 s	0.42 s
Konference	0.08 s	0.08 s	•
Lidský kontakt	0.16 s	0.45 s	•

Tabulka 3: Časy běhu vlastního programu

V druhé tabulce je běh algoritmu bez tvorby grafů. Místo převodu je zde pak uvedena délka načítání grafu do Gephi. Jak je vidět, tak pro emailovou komunikaci se nepovedlo vypočítat TCC pro okno velikosti jedné hodiny. Byla přesažena maximální velikost paměti programu Gephi.

	Načtení	TCC - 1 hodina	TCC - 1 den
Emailová komunikace	23.5 s	nedostatek paměti	52.1 s
Konference	7 s	1.2 s	•
Lidský kontakt	3.3 s	4.4 s	•

Tabulka 4: Časy běhu Gephi

10 Závěr

Komplexní sítě jsou sítě o milionech uzlů, které se mohou v čase měnit a vyvíjet. V této práci jsme se pokusili na malých sítích ukázat, jak se dají takové sítě popsat. Nejprve ve statické podobě a poté v dynamické podobě, kdy jsme se zaměřili na tzv. temporální centrality a change centralitu.

Podle definic z literatury jsme vypočítali všechny hodnoty centralit, které jsme reprezentovali pomocí grafu setříděných hodnot centralit u jednotlivých uzlů. Výsledkem pro každý uzel bylo jedno číslo, které mělo charakterizovat danou temporální centralitu pro specifikované časové řezy. Možnost popsat vývoj uzlu z období několika dnů, či let, je dosti malá. Jako zajímavější řešení je popsat chování uzlů v čase pomoci mezivýpočtů dané centrality. Tyto mezivýpočty byly vizualizovány pomocí heatmapy. Při správně zvolené velikosti okna jsme pak dokázali určit periodu, se kterou se mezivýpočty opakují. Dokázali jsme také určit uzly, které ve výsledku měly průměrnou hodnotu dané centrality, ale v některém z oken měli danou centralitu znatelně vyšší než jiné uzly. Dále jsme dokázali odhalit uzly, které v dané síti ukončily aktivitu a nebyly dále aktivní.

Temporální vlastnosti jsem implementovali jako plugin do nástroje Gephi i jako vlastní aplikaci. Ukázalo se, že Gephi není na analýzu dynamických vlastností příliš vhodný nástroj. Možnost analýzy dynamických datových kolekcí byla do Gephi přidána až později. Nepodařilo se tak tuto možnost dobře integrovat do stávajícího nástroje a vyskytuje se zde mnoho chyb, které nemáme možnost ovlivnit. V porovnání s vlastním nástrojem byla analýza pomalejší a paměťově více náročná. Samotný nástroj však byl optimalizován přímo pro daný vstup.

Pro výpočet těchto vlastností na větších sítích by tyto algoritmy, v této podobě, nebylo možné použít, jelikož jsou paměťově i výpočetně velmi náročné. Možným řešením by například mohlo být pozměnit definice. TCC bychom mohli počítat pouze do určité vzdálenosti a nemuseli tak zbytečně procházet celý časově uspořádaný graf. Je-li totiž cesta velmi dlouhá, výslednou centralitu to nakonec příliš nezmění.

Výpočet centralit samotných, pouze jako jediné číslo, nám nedá příliš velkou vypovídající hodnotu. Ukázalo se však, že zobrazování mezivýpočtů nám může odhalit velmi zajímavé chování některých uzlů. Při úpravě definic takovýchto vlastností a zaměřením se na jednotlivá okna, bychom pak mohli lépe analyzovat dynamické sítě a docílit tak zajímavých výsledků.

11 Reference

[1] Haggle network dataset – KONECT, April 2015.

[2] Hypertext 2009 network dataset – KONECT, April 2015.

[3] Manufacturing emails network dataset – KONECT, April 2015.

[4] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

[5] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[6] Heiko Böck. *The Definitive Guide to NetBeans^{TM} Platform 7*. Apress, 2011.

[7] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, (just-accepted):00–00, 2014.

[8] Paolo Federico, Jurgen Pfeffer, Wolfgang Aigner, Silvia Miksch, and Lukas Zenk. Visual analysis of dynamic networks using change centrality. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 179–183. IEEE Computer Society, 2012.

[9] Gephi. `http://gephi.github.io`. Dostupné: 1. 5. 2015.

[10] David Gilbert. The jfreechart class library. *Developer Guide. Object Refinery*, 7, 2002.

[11] F Harary and G Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–87, 1997.

[12] JFreeChart. `http://www.jfree.org/jfreechart/`. Dostupné: 1. 5. 2015.

[13] Hyoungshick Kim and Ross Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.

[14] Petr Kovář. Úvod do teorie grafů, 2012.

[15] Vito Latora and Massimo Marchiori. A measure of centrality based on network efficiency. *New Journal of Physics*, 9(6):188, 2007.

[16] ORACLE. `http://www.oracle.com/technetwork/java`. Dostupné: 1. 5. 2015.

[17] NetBeans Platform. `https://netbeans.org/features/platform/index.html`. Dostupné: 1. 5. 2015.

[18] John Tang, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Exploiting temporal complex network metrics in mobile malware containment. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–9. IEEE, 2011.

[19] Gephi tutoriál. http://gephi.github.io/users/. Dostupné: 1. 5. 2015.

[20] JFreeChart tutoriál. http://www.tutorialspoint.com/jfreechart/. Dostupné: 1. 5. 2015.

[21] Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6–20, 2003.

[22] Gephi wiki. https://github.com/gephi/gephi/wiki. Dostupné: 1. 5. 2015.